HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

MASTERARBEIT

Herr

Philipp Engler

Projektierung eines Mobile Devices zur Überwachung von Prozessen in Drahtlos-netzwerken

Mittweida, 2017

Institut für Technologie- und Wissenstransfer (ITWM)
in Kooperation mit der Fakultät Angewandte Computer- und Biowissenschaften

MASTERARBEIT

Projektierung eines Mobile Devices zur Überwachung von Prozessen in Drahtlosnetzwerken

Autor:
Herr

Philipp Engler

Studiengang:
Industrial Management

Seminargruppe:
ZM14-w1M

Erstprüfer:
Herr Professor Dr. rer. nat. Christian Hummert

Zweitprüfer:
Herr Professor Dr.-Ing. Uwe Schneider

Einreichung:
Mittweida, 11.01.2017

Verteidigung/Bewertung:
Mittweida, 2017

Institute for Technology and Knowledge Transfer
(ITWM) in cooperation with faculty of Applied Computer Sciences & Biosciences

MASTER THESIS

Creating a mobile device for wireless network monitoring

author:
Mr.

Philipp Engler

course of studies:
Industrial Management

seminar group:
ZM14w1-M

first examiner:
Mr. Professor Dr. rer. nat. Christian Hummert

second examiner:
Mr. Professor Dr.-Ing. Uwe Schneider

submission:
Mittweida, 11.01.2017

defense / evaluation:
Mittweida, 2017

Bibliografische Beschreibung:

Engler, Philipp:

Projektierung eines Mobile Devices zur Überwachung von Prozessen in Drahtlosnetzwerken. - 2017. - VI, 64 S.

Creating a mobile device for wireless network monitoring. - 2017. - VI, 64 p.

Mittweida, Hochschule Mittweida, University of Applied Sciences, Institut für Technologie- und Wissenstransfer (ITWM) in Kooperation mit der Fakultät Angewandte Computer- und Biowissenschaften, Masterarbeit, 2017

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Ziel und Zweck

Es soll ein sogenannter Wireless LAN Repeater, also ein Gerät, das die Reichweite der Funksignale eines Drahtlosnetzwerkes erweitert, entwickelt werden. Der Wireless LAN Repeater soll eine Verbindung zwischen dem Endgerät und dem Wireless LAN Access Point ermöglichen und den kompletten Netzwerkverkehr zwischen dem Endgerät und dem Wireless LAN Access Point auf eine Speicherkarte zur späteren Auswertung ablegen. Die Umgebung des Wireless LAN Repeaters wird in Abbildung 1 dargestellt.

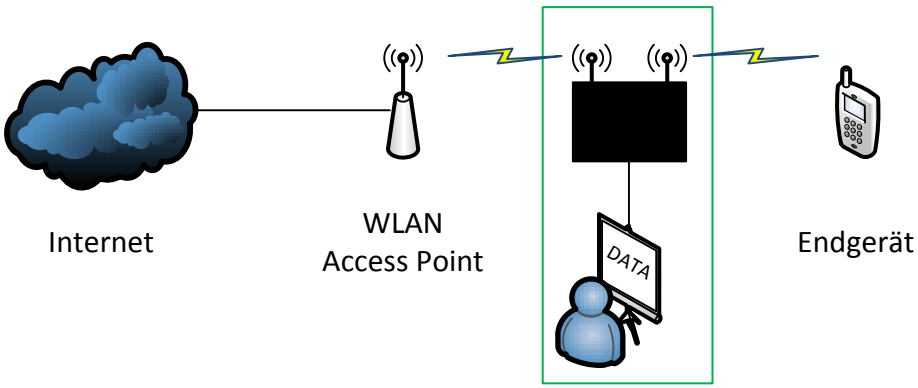

Abbildung 1: Übersicht des kompletten Aufbaus

Die gezeigten Bestandteile werden in Tabelle 1 kurz beschrieben.

Internet	Stellt das Internet dar.
WLAN Access Point	Stellt einen Wireless LAN Access Point dar, mit dem ein Endgerät eine Verbindung zum Internet herstellen kann.
WLAN Repeater	Stellt ein Gerät dar, das die Signale des Wireless LAN Access Point empfängt und an das Endgerät weiterleitet beziehungsweise die Signale des Endgerätes empfängt und an den Wireless LAN Access Point weiterleitet. Außerdem sollen die gesendeten Informationen auf einem Speichermedium gesichert werden.
Endgerät	Das Endgerät, das eine Verbindung zum Internet aufbaut.

Tabelle 1: Erläuterung der Bestandteile des kompletten Aufbaus

Neben der eigentlichen Funktion des Repeaters, der Weiterleitung der Signale vom Access Point an das Endgerät beziehungsweise vom Endgerät an den Access Point, soll der zu entwickelnde Repeater folgende Anforderungen erfüllen.

- Der Repeater soll unsichtbar arbeiten. Das bedeutet, dass der Repeater im Drahtlosnetzwerk nicht angemeldet werden muss. Er darf im Netzwerk nicht als Netzwerkgerät erscheinen und weder vom Access Point aus noch vom Endgerät aus sichtbar sein.
- Der Repeater soll die empfangenen und gesendeten Signale unverändert in die jeweilige Richtung weiterleiten.
- Der Repeater soll die empfangenen Signale auf einem Speichermedium – beispielsweise einer SD-Karte – für eine spätere Auswertung zwischenspeichern können.

In diesem Masterprojekt soll ein Teil des Wireless LAN Repeaters realisiert werden. Dabei handelt es sich um den in Abbildung 2 grün umrandeten Bereich, also die Kommunikation zwischen dem Endgerät und dem Wireless LAN Repeater und die Speicherung des Datenverkehrs auf ein Speichermedium zur späteren Auswertung.

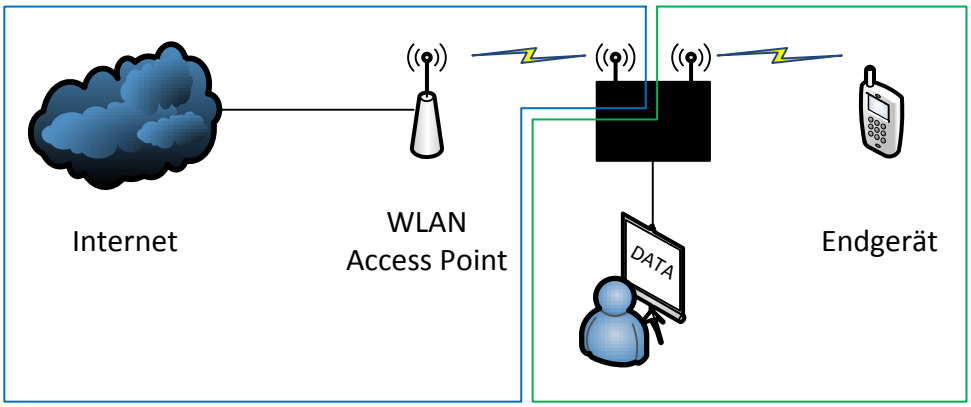

Abbildung 2: Übersicht des Projektes

Die dafür verwendende Hardware wird in Kapitel 1.2 beschrieben. Zur Funktionsfähigkeit wird neben dieser Hardware noch Software benötigt, die ebenfalls in diesem Masterprojekt entwickelt werden soll.

1.2 Hardware

1.2.1 Recheneinheit

Für den Aufbau eines solchen Wireless LAN Repeaters kann geeignete Hardware im Prinzip aus zwei Gruppen gewählt werden.

Einerseits existieren Geräte, die auf einer kleinen Platine einen vollwertigen Computer zur Verfügung stellen; zum anderen stehen sogenannte Embedded Systems – oder eingebettete Systeme – zur Verfügung. Im Folgenden werden beide Typen kurz charakterisiert.

1.2.1.1 Einplatinencomputer

Diese Geräte können mit einem Betriebssystem, in der Regel einer angepassten Version von Linux, betrieben werden. Der Prozessor basiert häufig auf der ARM-Architektur. Diese Prozessoren sind nicht so leistungsfähig wie Prozessoren für aktuelle Desktop-Computer oder Laptops, haben jedoch einen erheblich niedrigeren Leistungsbedarf und bedürfen bei normalem Einsatz keiner Kühlung.

Durch die Verwendung eines Betriebssystems benötigt der Computer vom Anlegen der Stromversorgung bis zur Einsatzbereitschaft mehr Zeit als ein Embedded System, da unter anderem das Betriebssystem geladen werden muss.

Über einen Grafikausgang kann ein Monitor angeschlossen werden. Eingabegeräte und weitere Hardware lassen sich häufig über die zur Verfügung stehenden USB-Buchsen anschließen.

Die Einplatinencomputer werden vielfältig eingesetzt. Die Einsatzgebiete reichen von Abspielgeräten für Bilder, Videos oder Musik über kleine Server im Netzwerk, zum Beispiel für Druckaufgaben, bis hin zu Zentralen für Hausautomatisierungssysteme.

Die bekanntesten Vertreter dieser Gruppe sind beispielsweise der Raspberry Pi [1] oder das Beagle Board [2].

1.2.1.2 Embedded Systems

Embedded Systems werden in der Regel ohne ein Betriebssystem betrieben. Auch die hier verwendeten Prozessoren basieren auf der ARM-Architektur, sind jedoch deutlich niedriger getaktet als die Prozessoren der Einplatinencomputer. Wie bei den Einplatinencomputern kann auch bei Embedded Systems häufig auf eine Kühlung verzichtet werden.

Für den Betrieb wird kein Betriebssystem benötigt, wodurch das Gerät sofort nach dem Anlegen der Betriebsspannung beziehungsweise dem Einschalten die Arbeit aufnimmt. Die Programme werden in den meisten Fällen mit Hilfe einer Programmiersprache aus der C-Familie erstellt und anschließend mittels Compiler und Linker in ein für das Zielsystem kompatibles Format gebracht.

Da es kein Betriebssystem gibt, besitzen diese Systeme meistens weder einen Grafikausgang noch die Möglichkeit, Eingabegeräte anzuschließen. Zusätzliche Hardware kann häufig über Stiftleisten angeschlossen werden.

Die Embedded Systems werden heute in einem Großteil der am Markt befindlichen Geräte verbaut. Beispielsweise werden sie in Waschmaschinen genutzt, um die Steuerung der

Waschprogramme und die Anzeige des Status mit Hilfe einer LED-Anzeige oder auf einem Bildschirm zu ermöglichen. Der Vorteil dieser Geräte liegt in der Variabilität der Einsatzgebiete und dem sehr niedrigen Energiebedarf.

Im Bereich der eingebetteten Systeme existiert eine große Anzahl an verschiedenen Systemen. Am bekanntesten sind hier die verschiedenen Arduino-Plattformen [3].

1.2.1.3 Auswahl der Recheneinheit

In diesem Projekt wird ein Embedded System verwendet. Dabei handelt es sich um den STM32 Nucleo [4] von STMicroelectronics [5]. In dieser Reihe existieren verschiedene Untertypen, die sich unter anderem im Packaging des verwendeten Mikrocontrollers und der Größe des vom Mikrocontroller verwendeten Programmspeichers unterscheiden. Die hier verwendete Version NUCLEO-F411RE besitzt den Mikrocontroller STM32F411RET6 [6] mit 64 Pins, bei dem es sich um einen ARM 32Bit Cortex-M4 Prozessor mit digitalem Signalprozessor und Gleitkommaeinheit handelt. Er besitzt eine Taktfrequenz von 100 Megahertz und stellt 512 Kilobyte Programmspeicher [7] sowie 128 Kilobyte SRAM zur Verfügung.

Der STM32 Nucleo kann über zwei verschiedene Stiftleisten mit zusätzlicher Hardware erweitert werden. Eine der Stiftleisten ist Arduino-UNO kompatibel, die andere ist zu ST Morpho kompatibel. Eine Beschreibung der Belegung beider Stiftleiten steht unter [8] zur Verfügung. In diesem Projekt werden an diese Stiftleisten das im Folgenden beschriebene Drahtlosnetzwerk-Modul sowie die ebenfalls beschriebene Platine zur Nutzung eines SD-Speichermediums angeschlossen.

1.2.2 Drahtlosnetzwerk-Modul

Als Drahtlosnetzwerk-Modul soll das ESP8266 [9] von Espressif Systems [10] verwendet werden.

Das Modul unterstützt laut Datenblatt [11] Drahtlosnetzwerke mit einer Funkfrequenz von 2,4 Gigahertz. Die Funknetze müssen den Standards IEEE 802.11b, IEEE 802.11g oder IEEE 802.11n entsprechen. Das Modul kann mit Netzwerken verbunden werden, die entweder nicht gesichert sind oder die Verschlüsselungstechnologien WEP, WPA oder WPA2 unterstützen, wobei der Sicherheitsschlüssel mit TKIP oder AES gesichert sein kann.

Zum Betrieb benötigt das Modul zwingend 3,3 Volt und ist nicht 5 Volt tolerant. Des Weiteren bietet der Adapter verschiedene Stromsparmodi [11].

1.2.3 SD Card Shield V4

Das in diesem Projekt verwendete SD Card Shield V4 von Seeed Development Limited [12] stellt einen Speicherkartenplatz für SD-Karten oder, über den mitgelieferten Adapter, für MicroSD-Karten zur Verfügung.

Die Verbindung zu Einplatinencomputern wird über die Arduino-UNO-kompatible Stiftleiste hergestellt. Der Zugriff auf die Speicherkarte erfolgt mittels Serial Peripheral Interface (SPI). Dieses Interface wird über eine zusätzliche Schnittstelle mit 6 PINs realisiert.

Über das Serial Peripheral Interface kann eine Kommunikation zwischen verschiedenen Hardwarekomponenten erfolgen. Entwickelt wurde das Serial Peripheral Interface von Motorola [13].

Der STM32 Nucleo stellt das Serial Peripheral Interface allerdings nicht über die Arduino-UNO Stiftleiste des SD Card Schield V4 zur Verfügung.

Als Speicherkarten können SD-, Micro SD- und SDHC-Karten mit einer maximalen Größe von 32 GB benutzt werden (vergleiche [12]).

Für die Nutzung des SD Card Shield auf dem Arduino steht dessen Arduino SD Library zur Verfügung. Für den STM32 Nucleo steht eine Bibliothek von mbed zur Verfügung.

1.3 Drahtlosnetzwerke

Auf Grund der immer größer werdenden Bedeutung von Laptops und Mobilgeräten seit Ende der neunziger Jahre wurde der Bedarf an einer kabellosen Netzwerkverbindung immer größer.

Bei den nachfolgend angegeben Geschwindigkeiten handelt es sich um theoretisch erreichbare Geschwindigkeiten. Durch die Übertragungsprotokolle und durch Hindernisse wie beispielsweise Wände und Decken können diese Werte in der Praxis nicht erreicht werden.

1.3.1 Entwicklung der Drahtlosnetzwerk-Technologie

Bereits Ende der sechziger Jahre wurde an der *University of Hawaii* mit dem ALOHAnet ein Drahtlosnetzwerk entwickelt und aufgebaut. Da Hawaii aus mehreren Inseln besteht, sollten mit dem Netzwerk die Inseln untereinander verbunden werden. Im Jahr 1971 wurde das Netzwerk mit einem Hauptstandort sowie sieben Nebenstellen in Betrieb genommen. Nachdem öffentliche Sponsoren die Förderung einstellten, musste der Betrieb im Herbst des Jahres 1976 eingestellt werden [14].

Der erste Standard für Drahtlosnetzwerke wurde im Jahr 1997 von der *Working Group for WLAN Standards* der IEEE [15] verabschiedet und trägt seitdem die Bezeichnung IEEE 802.11 [16]. Dieser Standard bildet auch heute noch die Grundlage für Drahtlosnetzwerke, wurde allerdings seit 1997 immer weiter aktualisiert und erweitert.

Im Jahr 1999 veröffentlichte die IEEE die Standards IEEE 802.11a [17] und IEEE 802.11b [18]. Die Unterschiede zwischen IEEE 802.11a und IEEE 802.11b liegen zum einen in den jeweils verwendeten Frequenzbändern, zum anderen in der maximal erreichbaren Übertragungsgeschwindigkeit. Während das IEEE 802.11a im 5 Gigahertz Frequenzbereich arbeitet, nutzt IEEE 802.11b den 2,4 Gigahertz Frequenzbereich. Im Gegensatz zum 5 Gigahertz Frequenzband konnte das 2,4 Gigahertz Frequenzband auch zur damaligen Zeit lizenzfrei genutzt werden. Dies führte dazu, dass IEEE 802.11b trotz der deutlich niedrigeren Übertragungsgeschwindigkeit von 11 Megabit je Sekunde im Vergleich zu IEEE 802.11a mit 54 Megabit je Sekunde eine größere Verbreitung fand.

Durch die im Jahr 2003 publizierte Erweiterung IEEE 802.11g [19] konnten auch im 2,4 Gigahertz Frequenzband eine Übertragungsgeschwindigkeit von 54 Megabit je Sekunde erzielt werden. Durch die Nutzung kompatibler Hardware war es möglich, sowohl IEEE 802.11b als auch IEEE 802.11g fähige Geräte nebeneinander zu betreiben. Dadurch erfolgte eine weite Verbreitung der IEEE 802.11g Netzwerktechnik.

Die 2009 verabschiedete Erweiterung IEEE 802.11n [20] ermöglicht es, dass Geräte sowohl im 2,4 Gigahertz Frequenzband als auch im 5 Gigahertz Frequenzband arbeiten können. Dies setzt allerdings Hardware voraus, die über einen Dualbandmodus verfügt. Der Standard IEEE 802.11n ermöglicht je nach verwendeter Hardware verschiedene Übertragungsgeschwindigkeiten. Diese unterschiedlichen Übertragungsgeschwindigkeiten sind möglich, weil der Standard die sogenannte MIMO-Technologie nutzt. Dabei steht MIMO für Multiple Input Multiple Output. Geräte, die MIMO unterstützen, besitzen mehrere Antennen. Somit können mehrere Datenübertragungskanäle zwischen zwei Geräten aufgebaut werden. Es wurden im Standard vier verschiedene Geschwindigkeitsklassen definiert (vergleiche Tabelle 2).

Anzahl der Antennen je Gerät		**Übertragungsgeschwindigkeit**
1	1x1 MIMO	150 MBit/s
2	2x2 MIMO	300 MBit/s
3	3x3 MIMO	450 MBit/s
4	4x4 MIMO	600 MBit/s

Tabelle 2: Übertagungsgeschwindigkeiten mit MIMO in IEEE 802.11n (in Anlehnung an [22])

Am weitesten verbreitet sind Geräte mit einer Übertragungsgeschwindigkeit von 300 Megabit je Sekunde (2x2 MIMO) und 450 Megabit je Sekunde (3x3 MIMO).

Auf der Basis von IEEE 802.11n wurde 2013 der Nachfolger IEEE 802.11ac [21] als Standard definiert. Im Gegensatz zu IEEE 802.11n, der beide Frequenzbänder unterstützt, arbeitet der Standard IEEE 802.11ac nur im 5 Gigahertz Frequenzband. Durch eine Optimierung der Modulation und eine Anhebung der maximalen Anzahl der Antennen auf acht konnte die Übertragungsgeschwindigkeit erhöht werden (vergleiche Tabelle 3).

Anzahl der Antennen je Gerät		**Kanalbreite**	**Übertragungsgeschwindigkeit**
1	1x1 MIMO	80 MHz	433 MBit/s
2	2x2 MIMO	80 MHz	867 MBit/s
3	3x3 MIMO	80 MHz	1300 MBit/s
4	4x4 MIMO	80 MHz	1700 MBit/s
8	8x8 MIMO	80 MHz	3464 MBit/s

Tabelle 3: Übertagungsgeschwindigkeiten mit MIMO in IEEE 802.11ac (in Anlehnung an [22])

Am Markt befinden sich zurzeit vor allem Geräte, die 1300 Megabit je Sekunde (3x3 MIMO) und 1700 Megabit je Sekunde (4x4 MIMO) unterstützen.

Bei dem Vergleich aktueller Hardware muss jedoch beachtet werden, dass die Hersteller häufig die Übertragungsrate des 2,4 Gigahertz Frequenzbandes und die Übertragungsrate des 5 Gigahertz Frequenzbandes addieren, um so mit höheren Übertragungsgeschwindigkeiten werben zu können. So besitzt beispielsweise ein dualbandfähiges Gerät mit einer beworbenen Übertragungsrate von 1750 Megabit je Sekunde ein 3x3 MIMO im 2,4 Gigahertz Frequenzband (IEEE 802.11n) und ein 3x3 MIMO im 5 Gigahertz Frequenzband (IEEE 802.11ac). Damit erreicht das Gerät im 2,4 Gigahertz Frequenzbereich 450 Megabit je Sekunde (vergleiche Tabelle 2) und im 5 Gigahertz Frequenzbereich 1300 Megabit je Sekunde (vergleiche Tabelle 3).

1.3.2 Überblick über verwendete Frequenzen

Auf Grund der Lizenzierungspflicht des 5 Gigahertz Frequenzbandes gab es zu Beginn der Drahtlosnetzwerke kaum Geräte, die dieses Frequenzband nutzten. Dies führte zu einer hohen Auslastung des 2,4 Gigahertz Frequenzbandes, was vor allem in dicht besiedelten Gebieten häufig zu Problemen führt. Erschwerend kommt hinzu, dass weitere Technologien, beispielsweise Bluetooth, das 2,4 Gigahertz Frequenzband nutzen.

Mittlerweile ist auch das 5 Gigahertz Frequenzband in vielen Ländern lizenzfrei nutzbar. Allerdings müssen zur Nutzung entweder dualbandfähige Drahtlosnetzwerkadapter und dualbandfähige Basisstationen vorhanden sein oder entsprechende Geräte beschafft werden. Viele preiswerte Geräte, beispielsweise einfache Tablet-PCs oder Mobiltelefone, unterstützen häufig nur das 2,4 Gigahertz Frequenzband.

Ein Nachteil des 5 Gigahertz Frequenzbands liegt in der im Vergleich zum 2,4 Gigahertz Frequenzband deutlich geringeren Reichweite. Die Funksignale des 5 Gigahertz Frequenzbands werden von Wänden und Decken deutlich stärker gedämpft als die Funksignale des 2,4 Gigahertz Frequenzbandes.

1.3.3 Verschlüsselungstechnologien

Mit der wachsenden Verbreitung der Drahtlosnetzwerke wurde die Sicherung gegen unerlaubten Zugriff immer wichtiger.

Das Verschlüsselungsprotokoll WEP (*Wired Equivalent Privacy*) ist in dem Standard IEEE 802.11 spezifiziert. Er sollte einen Schutz bieten, der dem Schutz kabelgebundener Netzwerke entspricht (vergleiche Übersetzung *Wired Equivalent Privacy*). Das Verschlüsselungsprotokoll WEP gilt seit einigen Jahren als unsicher und sollte nicht mehr verwendet werden. Um zu zeigen, wie leicht WEP zu überwinden ist, wurde 2007 an der TU Darmstadt von Erik Tews, Ralf-Philipp Weinmann und Andrei Pyshkin die Arbeit „*Breaking 104 bit WEP in less than 60 seconds*“ veröffentlicht [23].

Da WEP nicht mehr als sicher angesehen werden konnte, wurde 2003 von der WiFi Alliance das Verschlüsselungsprotokoll WPA (*WiFi Protected Access*) spezifiziert. Die WiFi Alliance ist ein im Jahr 1999 gegründeter Zusammenschluss von Herstellern. WPA beinhaltet eine Teilmenge des damals noch in Entwicklung befindlichen Standards IEEE 802.11i [24]. Als Basis dient bei WPA der bereits bei WEP genutzte RC4-Algorithmus. Jedoch wurden aus den Entwürfen des IEEE 802.11i wichtige Funktionen zur Erhöhung der Sicherheit entnommen. Heute wird vom Einsatz von WPA ebenfalls abgeraten. Anstelle von WPA sollte besser WPA2 verwendet werden, das nach dem Standard IEEE 802.11i implementiert wurde.

Die 2004 eingeführte Verschlüsselungstechnik WPA2 (*WiFi Protected Access Version 2*) wurde nach dem Standard IEEE 802.11i implementiert und löst den Vorgänger WPA ab. Die im Standard IEEE 802.11i veröffentlichten Maßnahmen definieren eine neue Netzwerkarchitektur namens Robust Security Network (RSN) und stellen den aktuellen Sicherheitsstandard für Drahtlosnetzwerke dar.

1.4 Angriffsszenarien auf Drahtlosnetzwerke

Der Zugriff auf ein Netzwerk mit Drahtlostechnologie erfordert keine physische Verbindung mit dem Netzwerk. Dies ist sowohl für den Nutzer des Netzwerkes als auch für Angreifer sehr komfortabel.

Insbesondere bei drahtlosnetzwerkfähigen Netzwerkroutern ist ebenfalls ein Angriff über die Internetverbindung möglich, im Folgenden werden jedoch zwei Angriffsszenarien betrachtet, bei denen kein physischer Zugriff auf das Netzwerk notwendig ist.

1.4.1 Abhören von Drahtlosnetzwerken

Damit der Verkehr in Drahtlosnetzwerken mitgehört werden kann, muss einerseits der Drahtlosnetzwerkadapter in einen speziellen Betriebsmodus versetzt werden und andererseits ein spezielles Programm verwendet werden.

Der spezielle Betriebsmodus ist der sogenannte Monitor-Mode [25]. In den Treibern für Windows wird dieser von den meisten Herstellern nicht unterstützt, unter Linux lässt sich dieser Modus bei den meisten Drahtlosnetzwerkadaptern aktivieren. Der Monitor-Mode sorgt dafür, dass die empfangenen Daten komplett und unverfälscht an das Betriebssystem weitergegeben werden. Befindet sich der Drahtlosnetzwerkadapter in diesem Modus, ist es ihm nicht möglich, sich mit einem Drahtlosnetzwerk zu verbinden.

Die spezielle Software ist eine Sniffing-Software. Dieses Programm empfängt die vom Drahtlosadapter weitergereichten Informationen und stellt diese anschließend zur weitergehenden Analyse und Auswertung bereit. Auf Grund der im Jahre 2007 in Kraft getretenen Strafrechtsänderungen zur „Bekämpfung der Computerkriminalität“ (§§ 202a, 202b, 202c, 202d StGB [26]) ist der Einsatz dieser Programme in Deutschland weitestgehend untersagt. Dieses Verbot ist jedoch nicht unumstritten, da die Software auch von Administratoren verwendet wird, um mögliche Angriffspunkte auf das Drahtlosnetzwerk aufzufinden [27].

Für Angreifer ist diese Angriffsart besonders vorteilhaft. Sie können auf diese Art Informationen zum Drahtlosnetzwerk erhalten und der Angriff läuft völlig transparent ab. Dies bedeutet, dass sich weder am Netzwerk selbst noch in vorhandenen Logdateien Hinweise zu dem Angriff finden lassen.

1.4.2 Manipulation von Drahtlosnetzwerken

Zur Manipulation eines Drahtlosnetzwerks stehen verschiedene Möglichkeiten zur Verfügung. Im Folgenden werden drei Möglichkeiten näher betrachtet.

1.4.2.1 Angriff mit Hilfe von Man-in-the-Middle

Um Zugriff auf Drahtlosnetzwerkgeräte zu erhalten, kann ein Drahtlosnetzwerk erzeugt werden, das entweder die SSID eines an der Stelle vorhandenen Netzwerks besitzt oder die SSID eines an einem öffentlichen Ort vorhandenen Drahtlosnetzwerks nutzt, mit dem das anzugreifende Gerät bereits verbunden war. In beiden Fällen arbeitet das angreifende Drahtlosnetzwerkgerät als Access Point.

Wird ein an der Stelle bestehendes Drahtlosnetzwerk simuliert, wird sich das Endgerät in der Regel immer mit der Basisstation verbinden, deren Signal am stärksten verfügbar ist. Dies bedeutet, dass das angreifende Gerät entweder eine leistungsfähige Sendeeinheit besitzen muss oder nahe am anzugreifenden Gerät platziert werden muss.

Bei der Simulation eines Drahtlosnetzwerks mit einem an einem öffentlichen Ort verfügbaren Drahtlosnetzwerk muss das aufgespannte Drahtlosnetzwerk die gleiche Sicherheitskonfiguration besitzen wie das ursprüngliche Drahtlosnetzwerk. Weiterhin muss das Endgerät mit diesem Netzwerk bereits verbunden gewesen sein und das Drahtlosnetzwerkprofil gespeichert worden sein. Ist dies der Fall, wird sich das Endgerät automatisch mit dem Drahtlosnetzwerk verbinden, sofern kein anderes dem Endgerät bekanntes Drahtlosnetzwerk verfügbar ist, das in den Einstellungen des Endgerätes höher priorisiert ist.

Ist das Endgerät verbunden, kann der Angreifer die Kommunikation abhören und zwischenspeichern. Ebenfalls ist es möglich, gesendete und empfangene Daten zu verändern.

1.4.2.2 Angriff auf vom Hersteller generierte Sicherheitsschlüssel

Die Drahtlosnetzwerkgeräte, die das Drahtlosnetzwerk zur Verfügung stellen, werden häufig mit einer Standardkonfiguration geliefert. Zu dieser Standardkonfiguration gehört ebenfalls, dass bereits eine SSID und ein Passwort eingestellt sind.

Vor allem im Privatkundenbereich kam es in den vergangenen Jahren immer wieder dazu, dass drahtlosnetzwerkfähige Internetrouter anfällig waren. Wurden nach der Einrichtung die SSID und das Passwort nicht geändert, konnten diese Drahtlosnetzwerkverbindungen mit verhältnismäßig geringem Aufwand kompromittiert werden.

Anhand der nicht geänderten SSID kann man herausfinden, um welches Gerät es sich handelt. Durch das Abhören des Drahtlosnetzwerks (vergleiche Kapitel 1.4.1) ist es möglich, die MAC-Adresse der Basisstation zu ermitteln. In einem nächsten Schritt kann mit einem geeigneten Programm das Passwort aus der MAC-Adresse ermittelt werden. Dies ist möglich, da einige Hersteller das vorkonfigurierte Passwort während der Produktion mit Hilfe eines Algorithmus unter Verwendung der MAC-Adresse berechnen (siehe [28], [29] und [30]).

Nun kann eine Verbindung zum Netzwerk hergestellt werden. Da häufig weitere Sicherheitslücken in den Internetroutern existieren oder das Standardpasswort für die Konfigurationsoberfläche nicht geändert wurde, kann der Angreifer so im schlimmsten Fall sogar die Kontrolle über den Internetrouter übernehmen – und damit auch die Kontrolle über alle im Netzwerk befindlichen Geräte.

1.4.2.3 Verwendung einer bekannten MAC-Adresse

Um die Sicherheit von Drahtlosnetzwerken weiter zu erhöhen, wurde früher dazu geraten, eine Liste der Geräte zu erstellen, die auf das Netzwerk zugreifen dürfen. Zur Identifizierung dieser Geräte wurde die MAC-Adresse der Drahtlosnetzwerkkarte verwendet.

Da die MAC-Adressen unverschlüsselt und in jedem Paket übertragen werden müssen, kann man durch Abhören des Drahtlosnetzwerks (vergleiche Kapitel 1.4.1) mindestens eine im Netzwerk zugelassene MAC-Adresse herausfinden.

Zwar werden MAC-Adressen fix vergeben, sie lassen sich aber spoofen. Somit ist es möglich, die MAC-Adresse zu ändern und somit den MAC-Adressen-Filter zu umgehen. Ein Beispiel dafür wird in Abbildung 3 gezeigt.

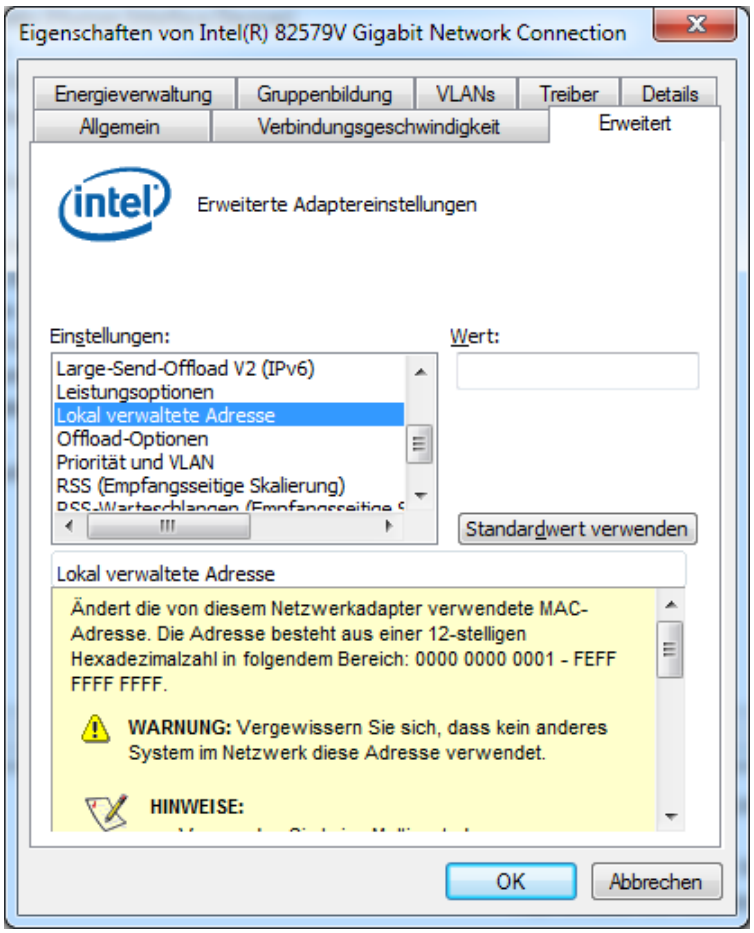

Abbildung 3: Ändern der MAC-Adresse eines Netzwerkadapters

2 Ergebnisse

Im Folgenden wird ein Überblick über den Aufbau gegeben und die entwickelte Alternative für den Promiscous-Mode genannt.

2.1 Aufbau der betriebsbereiten Hardware

Die Abbildung 4 zeigt den Aufbau und die Verbindung des STM32 Nucleo, des SD Card Shield V4 und der beiden Drahtlosnetzwerk-Module.

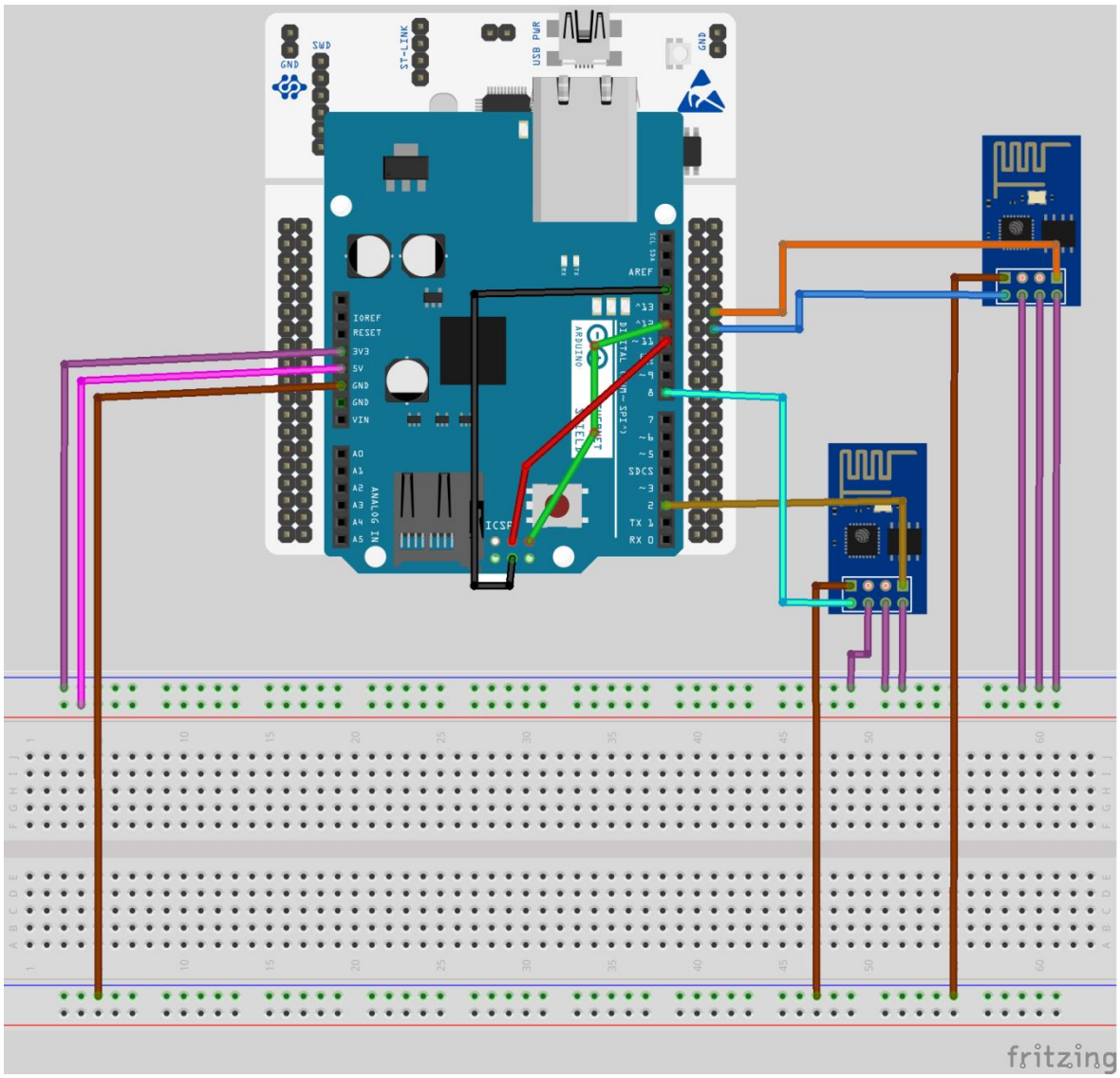

Abbildung 4: Aufbau und Verdrahtung der einzelnen Komponenten

Das SD Card Shield V4 wurde auf die Arduino-Steckleiste des STM32 Nucleo aufgesteckt. Da der STM32 Nucleo die SPI-Schnittstelle des SD Card Shield V4 nicht unterstützt, mussten von dieser Verbindungen zu den dazugehörigen PINs hergestellt werden.

Soll das SD Card Shield V4 nicht auf den STM32 Nucleo gesteckt werden, kann es alternativ angeschossen werden, indem eine Verbindung der PINs `5V`, `3V3` und `GND` zwischen dem STM32 Nucleo und dem SD Card Shield V4 hergestellt wird.

Die beiden Drahtlosnetzwerk-Module wurden über ein Steckboard mit dem STM32 Nucleo verbunden, um diese mit Strom zu versorgen, da die Anzahl der Anschlüsse ansonsten nicht ausreichend gewesen wäre.

Die Definition und Beschreibung der genutzten Anschlüsse und die Verbindung der PINs befinden sich im Kapitel 3.

2.2 Alternative für den Promiscous-Mode

Da es nicht möglich war, das Drahtlosnetzwerk-Modul ESP8266 in den Promiscous-Mode mit Hilfe von AT-Befehlen zu versetzen, wurde eine alternative Lösung entworfen. Diese erzeugt nicht das gleiche Ergebnis, zeigt allerdings, wie die Programmierung prinzipiell vorzunehmen ist. Die Umsetzung dieser Lösung wird in Kapitel 3.3.3.2 vorgestellt.

3 Methoden

In diesem Kapitel wird beschreiben, wie das Drahtlosnetzwerk-Modul und das SD Card Shield mit dem STM32 Nucleo verbunden werden und welche vorbereitenden Maßnahmen eventuell vorzunehmen sind, damit eine Kommunikation zwischen den Komponenten und dem STM32 Nucleo ermöglicht wird. Mit Hilfe kurzer Beispiele wird die Nutzung der Komponenten am STM32 Nucleo verdeutlicht.

3.1 Ansteuerung des Drahtlosnetzwerk-Moduls ESP8266

Für das Drahtlosnetzwerk-Modul ESP8266 stehen verschiedene Firmwareversionen zur Verfügung, die untereinander nicht kompatibel sind. Der Unterschied zwischen den verschiedenen Firmwareversionen besteht vor allem darin, wie – also mit welchen Befehlen oder mit welchem Befehlssatz – das Drahtlosnetzwerk-Modul angesprochen werden kann. Damit das Drahtlosnetzwerk-Modul mit dem STM32 Nucleo genutzt werden kann, muss das Modul über eine AT-fähige Firmware verfügen. Kann das Drahtlosnetzwerk-Modul nicht mit dem STM32 Nucleo kommunizieren, muss die Firmware des Moduls aktualisiert werden. Dieser Vorgang wird in Kapitel 3.1.1 beschrieben.

Das Drahtlosnetzwerk-Modul ist nicht PIN-kompatibel zum STM32 Nucleo. Das bedeutet, dass das Drahtlosnetzwerk-Modul über eine individuelle Verdrahtung mit dem STM32 Nucleo verbunden werden muss. Die verwendete Konfiguration wird im Kapitel 3.1.2 betrachtet.

Nachddem die Verbindung der Hardware hergestellt wurde, wird zwischen dem Drahtlosnetzwerk-Modul und dem STM32 Nucleo eine serielle Schnittstelle für die Kommunikation zwischen beiden Hardwarekomponenten aufgebaut. Dieses Vorgehen wird im Kapitel 3.1.3 beschrieben.

Für einen Test des Drahtlosnetzwerk-Moduls kann eine Steuerung durch das Senden der AT-Befehle über eine definierte serielle Schnittstelle erfolgen. Dieses Vorgehen wird in Kapitel 3.1.4 beschrieben. Für das Projekt wird eine Bibliothek WeeESP8266 benutzt, die von ITEAD STUDIO zur Verfügung gestellt wird.

3.1.1 Aktualisierung der Firmware des Drahtlosnetzwerk-Moduls

Das zur Verfügung stehende Drahtlosnetzwerk-Modul beherrscht im Auslieferungszustand keine AT-Befehle. Dieser Befehlssatz ist allerdings zum Betrieb mit dem STM32 Nucleo notwendig. Damit das Drahtlosnetzwerk-Modul den AT-Befehlssatz unterstützen

kann, wird vom Hersteller des Drahtlosnetzwerk-Moduls auf GitHub [31] eine entsprechend modifizierte Firmware zur Verfügung gestellt.

Damit das Drahtlosnetzwerk-Modul mit der AT-fähigen Firmware aktualisiert werden kann, müssen folgenden Voraussetzungen erfüllt sein:

- USB zu Seriell Adapter, der 3,3 Volt zur Verfügung stellen kann,
- Drahtlosnetzwerk-Modul ESP8266,
- Software zur Aktualisierung des Drahtlosnetzwerk-Moduls,
- Firmware-Dateien und
- Windows-Computer.

Auf ARMmbed stellt Sebastian Schocke eine Anleitung [32] zur Verfügung, in der er das Vorgehen beschreibt sowie die benötigten Dateien verlinkt.

Der zum Update genutzte USB zu Seriell Adapter verfügt über sechs Anschlusspins. Diese werden wie folgt mit den 6 PINs des Drahtlosnetzwerk-Moduls verbunden:

PIN des WLAN-Moduls	Bedeutung	PIN des USB zu Seriell Adapters
GND	Anschluss für Masse.	GND
GPIO0	Anschluss des General Purpose I/O.	GND
RX	Anschluss für Receive.	RXD
TX	Anschluss für Transmit.	TXD
CH_PD	Anschluss für Chip Powerdown. Versetzt den Chip in einen Stromsparmodus wenn an GND angeschlossen.	5V
VCC	Anschluss für die Betriebsspannung (Achtung: nicht tolerant zu 5 Volt).	5V

Tabelle 4: Beschaltung für das Firmwareupdate

Die Anschlusspins CTS und DTR des USB zu Seriell Adapters sowie die Anschlusspins GPIO2 und RESET des Drahtlosnetzwerk-Moduls werden nicht genutzt.

Trotz der Beschriftung des Anschlusspins 5V stehen an diesem 3,3 Volt zu Verfügung, wenn der Schalter auf der Oberseite auf 3V3 geschaltet wurde. Zur Sicherheit kann diese Spannung mit einem Voltmeter kontrolliert werden, bevor das Drahtlosnetzwerk-Modul angeschlossen wird. An dieser Stelle wird nochmals darauf hingewiesen, dass das Draht-

losnetzwerk-Modul ESP8266 nicht an eine Versorgungsspannung von 5 Volt angeschlossen werden darf.

Nun muss der USB zu Seriell Adapter mit einem Computer verbunden werden. Das Firmwareupdateprogramm steht ausschließlich für Microsoft Windows zur Verfügung. Anschließend muss der für den USB zu Seriell Adapter geeignete Treiber installiert werden. Wurde dieser nicht mitgeliefert, so können im Gerätemanager die Vendor ID (*VID*) und die Product ID (*PID*) des Adapters ausgelesen werden (Abbildung 5) und danach beim entsprechenden Hersteller nach dem passenden Treiber gesucht werden.

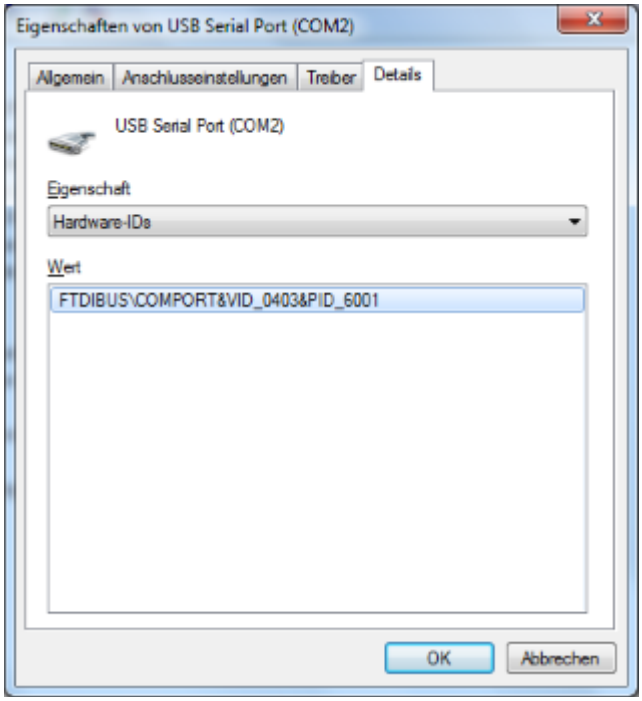

Abbildung 5: Anzeige VID und PID

Die Firmware für das Drahtlosnetzwerk-Modul wird vom Hersteller selbst auf GitHub zur Verfügung gestellt [31]. Sebastian Schocke stellt diese ebenfalls in einem Download mit dem dazugehörigen Frimwareupdateprogramm als ZIP-Archiv bereit [33].

Das von Espressif zur Verfügung gestellte Firmwarepaket beinhaltet die nachfolgend genannten fünf Dateien:

1. `blank.bin`
2. `boot_v1.1.bin`
3. `esp_init_data_default.bin`
4. `user1.bin`
5. `readme.txt`

Dabei beinhalten die Dateien eins bis vier die eigentliche Firmware, die Datei fünf ist die Readme-Datei, in der Informationen zum Updatevorgang enthalten sind.

In dem ZIP-Archiv von Sebastian Schocke sind neben den oben genannten fünf Dateien zwei zusätzliche Dateien (`XTCOM_UTIL.exe` und `XTCOM_API.dll`) enthalten. Dabei handelt es sich um das Firmwareupdateprogramm sowie eine dazugehörige Anwendungserweiterung.

Das Updateprogramm wird mit einem Doppelklick auf die Datei `XTCOM_UTIL.exe` gestartet. Es erscheint das Programmfenster (siehe Abbildung 6).

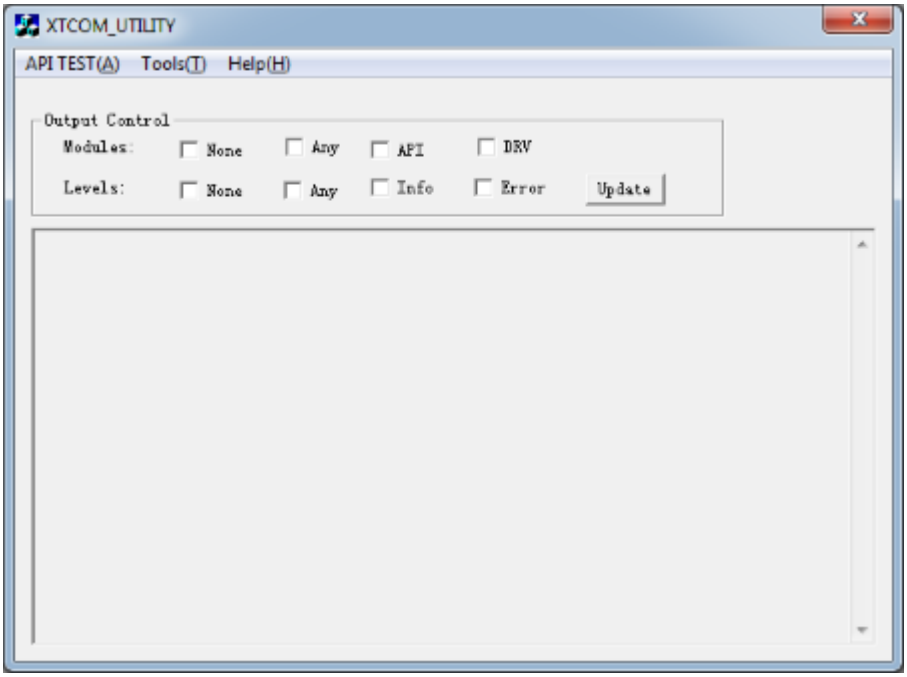

Abbildung 6: Firmwareupdateprogramm

Im nächsten Schritt muss der serielle Port konfiguriert werden. Dazu wird unter TOOLS (T) | CONFIG DEVICE der CONFIG DEVICE-Dialog (vergleiche Abbildung 7) geöffnet und dort die serielle Schnittstelle des USB zu Seriell Adapters eingestellt.

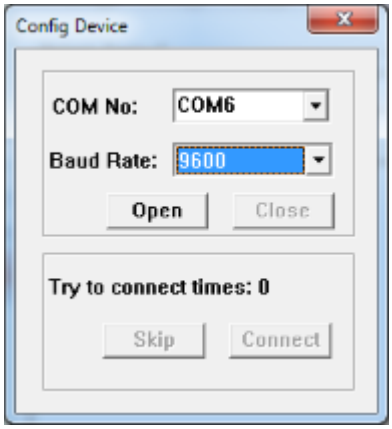

Abbildung 7: Config Device Fenster

Im Feld COM NO wird der entsprechende Port ausgewählt. Es können nur die seriellen Schnittstellen `COM1` bis `COM6` benutzt werden. Sollte der Adapter eine andere Schnittstelle

benutzen, muss diese im Gerätemanager auf einen unterstützten Port geändert werden (vergleiche Abbildung 8). Als Baudrate muss im Feld BAUD RATE `9600` eingestellt werden.

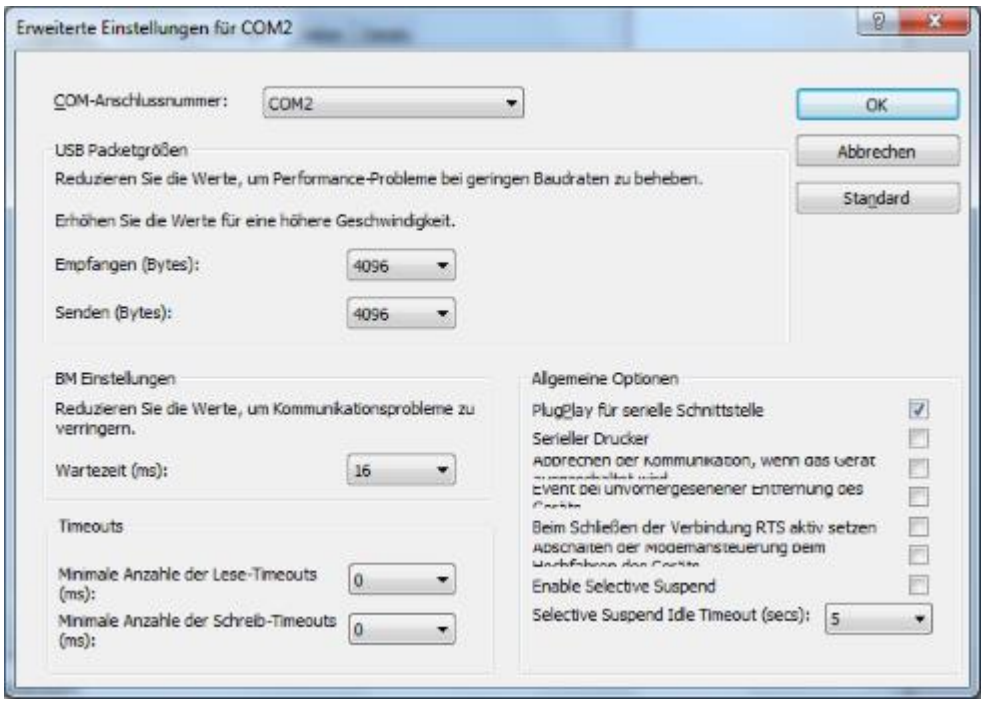

Abbildung 8: Ändern des COM-Ports

Anschließend wird die serielle Schnittstelle mit einem Klick auf OPEN geöffnet. Danach kann ein Verbindungstest mit einem Klick auf CONNECT ausgeführt werden. Falls keine Verbindung zu Stande kommen, sollte die Verbindung der Anschlusspins überprüft werden. Unter Umständen kann es notwendig sein, die `RX` und `TX`-Anschlüsse zu vertauschen. Wurde der Vorgang erfolgreich beendet, kann das DEVICE CONFIG-Fenster geschlossen werden.

Der eigentliche Firmwareupdateprozess wird im FLASH IMAGE DOWNLOAD-Dialog (vergleiche Abbildung 9) durchgeführt, der mit einem Klick auf API-TEST (A) | (4) FLASH IMAGE DOWNLOAD geöffnet wird.

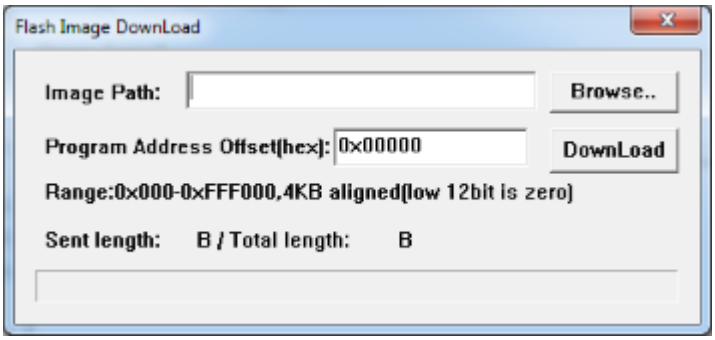

Abbildung 9: Flash Image Download Fenster

Im Feld IMAGE PATH werden nacheinander die vier in der in Tabelle 5 genannten Dateien in der angegebenen Reihenfolge an die dort angegebene Adresse (PROGRAMM ADDRESS OFFSET) geschrieben.

Dazu werden diese mittels BROWSE… ausgewählt und der entsprechende PROGRAM ADDRESS OFFSET eingegeben. Danach wird die entsprechende Datei mit einem Klick auf DOWNLOAD an die entsprechende Stelle im Speicher des Drahtlosnetzwerk-Moduls geschrieben.

Reihenfolge	Dateiname	Program Address Offset
1	`boot_v1.1.bin`	`0x00000`
2	`user1.bin`	`0x01000`
3	`esp_init_data_default.bin`	`0x7C000`
4	`blank.bin`	`0x7E000`

Tabelle 5: Firmwaredateien und deren Program Address Offset

Der in Tabelle 5 genannte Program Address Offset für jede einzelne Datei befindet sich in der Datei `readme.txt`. Wird eine neuere Firmware oder eine Firmware von einer anderen Quelle verwendet, müssen die in der entsprechenden Readme-Datei angegebenen Adressen für den Program Address Offset verwendet werden.

Nach dem Übertragen jeder einzelnen Datei muss das Firmwareupdateprogramm komplett geschlossen und der USB zu Seriell Adapter vom Computer getrennt werden. Das Drahtlosnetzwerk-Modul wird dadurch stromlos und startet bei der Wiederkehr der Versorgungsspannung mit den bereits aktualisierten Teilen der Firmware.

3.1.2 Anschluss des Drahtlosnetzwerk-Moduls an den STM32 Nucleo

Das Drahtlosnetzwerk-Modul ESP8266 besitzt acht PINs zum Anschluss an eine beliebige Hardware. Von der Oberseite aus betrachtet, stehen folgende PINs zur Verfügung:

PIN 1	PIN 2	PIN 3	PIN 4
GND	GPIO2	GPIO0	RX
TX	CH_PD	RESET	3V3
PIN 5	PIN 6	PIN 7	PIN 8

An dieser Stelle wird nochmals darauf hingewiesen, dass das Drahtlosnetzwerk-Modul ESP8266 nicht an eine Versorgungsspannung von 5 Volt angeschlossen werden darf.

Dabei besitzen die PINs folgende Bedeutung und werden an die angegebenen PINs des STM32 Nucleo angeschlossen:

PIN des WLAN-Moduls	Bedeutung	PIN des STM32 Nucleo
GND	Anschluss für Masse.	GND
GPIO2	Anschluss des General Purpose I/O 2.	n/a
GPIO0	Anschluss des General Purpose I/O 0.	n/a
RX	Anschluss für Receive.	RX-PIN (D2 oder PA_12)
TX	Anschluss für Transmit.	TX-PIN (D8 oder PA_11)
CH_PD	Anschluss für Chip Powerdown. Versetzt den Chip in einen Stromsparmodus wenn an GND angeschlossen.	+3V3
RESET	Anschluss für das Reset-Signal.	+3V3
VCC	Anschluss für die Betriebsspannung (Achtung: nicht tolerant zu 5V).	+3V3

Tabelle 6: Verbindung zwischen Drahtlosnetzwerk-Modul und STM32 Nucleo

Die beiden General Purpose PINs GPIO0 und GPIO2 werden für bestimmte Funktionen, etwa für ein Firmwareupdate, genutzt und müssen im normalen Betrieb nicht angeschlossen sein.

Bei dem Anschluss der RX und TX-PINs des Drahtlosnetzwerk-Moduls an den STM32 Nucleo muss darauf geachtet werden, dass entweder der serielle Anschluss Serial1 (PINs D2 und D8), oder der serielle Anschluss Serial6 (PINs PA_11 und PA_12) verwendet werden kann. Eine Überkreuzbelegung der beiden seriellen Anschlüsse ist nicht möglich.

Der Anschluss Serial2 (PINs D0 und D1) kann nicht für den Anschluss von Hardware verwendet werden, solange STLink Debug zum Beispiel für die Anzeige von Statusinformationen über die USB-Schnittstelle verwendet wird. Eine Nichtnutzung von STLink Debug reicht nicht aus, um den Anschluss Serial2 nutzen zu können. Soll der Anschluss Serial2 genutzt werden, müssen zwei Jumper umgesteckt sowie zwei Lötbrücken geöffnet werden.

Wird der PIN CH_PD auf GND gelegt, wird das Drahtlosnetzwerk-Modul in einen Schlafmodus versetzt, der beendet wird, wenn der PIN CH_PD auf +3V3 gelegt wird.

Der Reset des Drahtlosnetzwerk-Moduls kann über den PIN RESET eingeleitet werden. Dazu muss RESET kurz auf GND und anschließend wieder auf +3V3 gelegt werden.

3.1.3 Aufbau einer seriellen Verbindung zum Drahtlosnetzwerk-Modul

Die serielle Verbindung zwischen dem STM32 Nucleo und dem Drahtlosnetzwerk-Modul wird mit Hilfe des folgenden Befehls aufgebaut:

```
Serial esp(D8, D2);
```

Dadurch wird eine serielle Verbindung mit dem Namen esp definiert. Im oben abgebildeten Beispiel wird der serielle Anschluss Serial1 verwendet. Dabei wird als Sende-PIN der PIN D8 und als Empfangs-PIN der PIN D2 verwendet.

Im Folgenden wird das Vorgehen zum Senden von Befehlen an das Drahtlosnetzwerk-Modul betrachtet. Das nachfolgende Beispiel sendet an das Drahtlosnetzwerk-Modul einen Befehl zum Zurücksetzten des Moduls.

```
esp.printf("AT+RST\r\n");
```

Die Antwort des Drahtlosnetzwerk-Moduls kann durch die folgende Funktion ausgewertet werden. Dabei wird überprüft, ob die Zeichenkette OK in der Antwort enthalten ist und danach entweder true oder false zurückgegeben.

```
bool read(void)
{
      char buffer[20];

      while (esp.readable())
      {
            esp.scanf("%s", &buffer);
      }

      if(strcmp(buffer, "\r\nOK")==0)
      {
            return true;
      }
      else
      {
            return false;
      }
}
```

Da die Bearbeitung der AT-Befehle durch das Drahtlosnetzwerk-Modul einige Zeit in Anspruch nehmen kann, verzögert sich auch die Antwort, die das Drahtlosnetzwerk-Modul zurücksendet. Aus diesem Grund kann es eventuell nötig sein, dass mittels

```
wait(10);
```

auf die Antwort des Drahtlosnetzwerk-Moduls gewartet wird. Bei dem eingesetzten Befehl handelt es sich um ein Warten des Programms, bei dem die Abarbeitung für die übergebene Zeit unterbrochen wird. Sollte das Drahtlosnetzwerk-Modul vor dem Ablauf der übergebenen Zeit fertig werden, kann das Warten nicht abgebrochen werden.

3.1.4 Steuerung des Drahtlosnetzwerk-Moduls

Das Drahtlosnetzwerk-Modul ESP8266 lässt sich mit Hilfe von sogenannten AT-Befehlen [34] steuern. Bei dem AT-Befehlssatz handelt es sich um einen Befehlssatz, der ursprünglich zur Steuerung von Modems genutzt wurde und von der Firma Hayes Communication entwickelt und später als Standard definiert wurde.

Das Drahtlosnetzwerk-Modul ESP8266 erwartet die AT-Befehle im folgenden Format:

```
AT+BEFEHL\r\n
```

Beispielsweise wird mit dem unten stehenden Befehl ein Reset des Drahtlosnetzwerk-Moduls ausgelöst.

```
AT+RST\r\n
```

Damit das Drahtlosnetzwerk-Modul konfiguriert werden kann, können auch Werte übergeben werden, die durch das Drahtlosnetzwerk-Modul gelesen und umgesetzt werden. Im Folgenden wird der WiFi-Modus des Drahtlosnetzwerk-Moduls auf die Funktion „Router“ gesetzt.

```
esp.printf("AT+CWMODE=1\r\n");
```

Nach einem ausgeführten AT-Befehl antwortet das Drahtlosnetzwerk-Modul entweder mit einem OK oder mit einer abweichenden Zeichenkette.

Mit Hilfe der AT-Befehle kann auch der aktuelle Zustand des Drahtlosnetzwerk-Moduls ausgelesen werden. Mit dem folgenden Befehl kann beispielsweise der zurzeit vom Drahtlosnetzwerk-Modul genutzte WiFi-Modus ausgelesen werden.

```
esp.printf("AT+CWMODE?\r\n");
```

In dem oben abgebildeten Beispiel wird entweder eine `0`, eine `1` oder eine `2` zurückgegeben, je nachdem in welchem WiFi-Modus sich das Drahtlosnetzwerk-Modul befindet.

3.2 Ansteuerung des SD Card Shield V4

Das SD Card Shield V4 besitzt eine Arduino-UNO kompatible Steckleiste. Über diese wird das SD Card Shield V4 vom STM32 Nucleo mit Strom versorgt. Für die Kommunikation mit der SD Karte wird eine weitere Steckleiste mit 6 PINs verwendet. Wie diese PINs mit dem STM32 Nucleo verbunden werden, wird in Kapitel 3.2.1 beschrieben.

Nachdem die Verbindung der Hardware hergestellt wurde, wird zur Kommunikation zwischen dem SD Card Shield V4 und dem STM32 Nucleo eine SPI-Verbindung zur Kommunikation zwischen beiden Hardwarekomponenten aufgebaut. Dieses Vorgehen wird im Kapitel 3.2.2 beschrieben.

Die Steuerung des SD Card Shield V4 erfolgt über eine fertige Bibliothek, die unter anderem eine Implementierung des FAT-Dateisystems beinhaltet und somit das Lesen und Schreiben von Daten beherrscht. Die Verwendung dieser Bibliothek wird in Kapitel 3.2.3 beschrieben.

3.2.1 Anschluss des SD Card Shield V4 an den STM32 Nucleo

Das SD Card Shield V4 wird in die Arduino-UNO kompatible Steckleiste des STM32 Nucleo gesteckt. Dadurch wird das SD Card Shield V4 vom STM32 Nucleo mit Strom versorgt. Weitere Arduino-UNO kompatible Komponenten können auf das SD Card Shield V4 gesteckt werden.

Weitere sechs PINs befinden sich im unteren Bereich des SD Card Shield V4. Dabei handelt es sich um das Serial Peripheral Interface. Dieses wird zur Kommunikation mit der SD-Karte benötigt.

Bei dem Serial Peripheral Interface handelt es sich um ein Bussystem, das auch unter der Bezeichnung Microwire bekannt ist. Für die Datenübertragung werden die drei in der folgenden Tabelle 7 benannten Leitungen benötigt.

Bezeichnung des PINs	Bedeutung	Beschreibung
`MOSI`	*Master Out Slave In*	Überträgt die Daten vom Master zum Slave

Bezeichnung des PINs	Bedeutung	Beschreibung
MISO	*Master In Slave Out*	Überträgt die Daten vom Slave zum Master
SCK	*Serial Clock*	Synchronisierungstakt

Tabelle 7: Leitungen für die Datenübertragung über das SPI

Um in einem Bussystem mit mehreren Slaves einen bestimmten Slave ansprechen zu können, wird zusätzlich für jeden Slave ein so genanntes *Chip Select* (CS) Signal benötigt. Durch das Chip Select-Signal ist es möglich, dass der Master einen bestimmten Slave auswählen kann und somit nur dieser Slave auf die Kommunikation mit dem Master reagiert. Einen Überblick über das Serial Peripheral Interface steht bei Arduino auf der Webseite [35] oder auf der Webseite microcontroller.net [36] zur Verfügung.

Das Serial Peripheral Interface des SD Card Shield V4 ist kompatibel zu dem auf der Webseite [35] beschriebenen Anschluss. Von der Oberseite des SD Card Shield V4 aus betrachtet stehen am Serial Peripheral Interface folgende PINs zur Verfügung:

PIN 5	PIN 3	PIN 1
RESET	SCK	MISO
GND	MOSI	VCC
PIN 6	PIN 4	PIN 2

Dabei besitzen die PINs folgende Bedeutung und werden an die angegebenen PINs des STM32 Nucleo angeschlossen:

PIN des SD Card Shield V4	Bedeutung	PIN des STM32 Nucleo
MISO	Anschluss für Master In Slave Out.	D12
VCC	Anschluss für die Betriebsspannung.	n/a
SCK	Anschluss für Serial Clock.	D13
MOSI	Anschluss für Master Out Slave In.	D11
RESET	Anschluss für das Reset-Signal.	n/a
GND	Anschluss für Masse.	n/a

Tabelle 8: Verbindung zwischen SD Card Shield V4 und STM32 Nucleo

Die PINs VCC, RESET und GND werden im Betrieb mit dem STM32 Nucleo nicht angeschlossen.

Der Anschluss SCK stellt ein Taktsignal zur Verfügung, um die Kommunikation der Hardwarekomponenten über die serielle Schnittstelle zu synchronisieren. Dieses Taktsignal muss am Anschluss D13 angeschlossen werden.

Über den PIN MISO sendet der Slave seine Daten zum Master. Dieser PIN wird mit dem PIN D12 des STM32 Nucleo verbunden.

Der Master sendet die Daten über den Anschluss MOSI an den Slave. Dazu muss der Anschluss MOSI mit dem Anschluss D11 des STM32 Nucleo verbunden werden.

3.2.2 Aufbau einer SPI-Verbindung zum SD Card Shield V4

Damit von der SD Karte gelesen und darauf geschrieben werden kann, muss einerseits über das Serial Peripheral Interface eine Verbindung hergestellt werden und andererseits auf das Dateisystem zugegriffen werden können.

Für den Zugriff auf SD-Karten über das Serial Peripheral Interface stellt ARMmbed die Bibliothek SDFileSystem [37] zur Verfügung. Um diese Bibliothek nutzen zu können, muss diese importiert werden.

```
#include "SDFileSystem.h"
```

Anschließend kann die Verbindung über das Serial Peripheral Interface zwischen dem STM32 Nucleo und dem SD Card Shield V4 mit Hilfe des folgenden Befehls aufgebaut werden.

```
SDFileSystem sd(D11, D12, D13, D4, "sd");
```

Dadurch wird ein Serial Peripheral Interface mit dem Mountpunkt sd definiert. Im oben abgebildeten Beispiel werden die in Tabelle 8 genannten Anschlüsse verwendet. Zusätzlich wird der PIN D4 benutzt, damit das *Chip Select* Signal für die SD Card erzeugt werden kann.

3.2.3 Nutzung des SD Card Shield V4

Da auf dem STM32 Nucleo kein Betriebssystem vorhanden ist, kann sowohl für den Zugriff auf die SD-Karte als auch für das Dateisystem nicht auf Betriebssystemkomponenten zugrückgegriffen werden. Eine Eigenentwicklung – vor allem für das genutzte FAT-Dateisystem – würde den Rahmen des Masterprojektes sprengen. Aus diesem Grund wird auf eine Bibliothek von ARMmbed zurückgegriffen.

Die im Projekt genutzte Bibliothek SDFileSystem [37] übernimmt den Zugriff auf das FAT-Dateisystem der SD-Karte und ermöglicht es, mit den bekannten Funktionen aus der Standard-IO-Bibliothek `stdio` der Programmiersprache C auf die SD-Karte zu schreiben und von der SD-Karte zu lesen.

3.2.3.1 Handling von Dateien auf der SD-Karte

Das Anlegen und Öffnen von Dateien auf der SD-Karte wird mit Hilfe des folgenden Quelltextes ermöglicht.

```
FILE *fp = fopen("/sd/test.txt", "a+");
```

Mit der Funktion `fopen` können Dateien in verschiedenen Modi geöffnet oder erzeugt werden. In dem hier betrachteten Beispiel werden der Funktion `fopen` die zwei Argumente „`/sd/test.txt`“ und „`a+`“ übergeben. Bei dem ersten Argument handelt es sich um die Datei sowie den Pfad zur Datei. Das zweite Argument gibt an, dass die Datei `text.txt` zum Schreiben geöffnet oder, wenn sie noch nicht existiert, erzeugt werden soll. Sollte die Datei schon existieren, werden alle darin enthaltenen Informationen gelöscht und durch den einzufügenden Text ersetzt.

Wurde die Datei erfolgreich geöffnet oder erstellt, gibt die Funktion `fopen` einen Zeiger auf die Datei, den sogenannten File Pointer (`fp`), zurück. Im Fehlerfalle wird der Wert *`NULL`* zurückgegeben.

Eine Beschreibung der Funktion `fopen` wird von der IEEE und der „The Open Group“ auf der gemeinsam betriebenen Webseite opengroup.org [38] bereitgestellt. Ist eine neuere Version der Beschreibung verfügbar, wird im Kopf der Seite darauf hingewiesen und auf diese verlinkt.

Nachdem eine geöffnete Datei manipuliert wurde, muss diese geschlossen werden. Dies wird mit Hilfe des folgenden Quelltextes ermöglicht.

```
fclose(fp);
```

Zum Schließen einer Datei erwartet die Funktion `fclose` lediglich den Zeiger auf die Datei, die geschlossen werden soll.

Werden Daten, beispielsweise eine Zeichenkette wie in Kapitel 3.2.3.2 gezeigt, in den File Pointer geschrieben, kann nicht sichergestellt werden, dass diese Änderungen sofort in die Datei übernommen werden und somit persistent auf dem Datenträger gespeichert werden. Diese werden in einem Puffer zwischengespeichert und dadurch eventuell erst später in die Datei geschrieben. Wird die Datei nicht geschlossen, können diese Änderungen verloren gehen, so lange sich diese ausschließlich in dem Puffer befinden. Durch das Schließen werden eventuell noch nicht geschriebene Daten in die Datei geschrieben.

Beim Lesen von Dateien muss beachtet werden, dass Daten, die aus einer Datei eingelesen aber noch nicht verarbeitet wurden, nach dem Schließen nicht mehr zur Verfügung stehen.

Eine Beschreibung der Funktion `fclose` wird auf der Webseite opengroup.org [39] bereitgestellt.

3.2.3.2 Schreiben auf die SD-Karte

Das Schreiben in eine auf der SD-Karte vorhandene Datei wird mit Hilfe des folgenden Quelltextes ermöglicht.

```
if (fp != NULL)
{
      fprintf(fp, "Testtext");
      fclose(fp);
}
else
{
      //Fehlerbehandlung keine Datei/SD-Karte
}
```

Nachdem der Zeiger auf die Datei mit `fopen` geöffnet wurde, kann mit Hilfe verschiedener Funktionen, beispielsweise der Funktion `fprintf`, eine Zeichenkette in die Datei geschrieben werden. Dazu wird der Funktion `fprintf` als erstes Argument der Zeiger auf die Datei und als zweites Argument die einzufügende Zeichenkette übergeben.

Die Funktion `fprintf` gehört zu einer Funktionsgruppe. Eine Beschreibung dieser Funktionsgruppe wird auf der Webseite opengroup.org [40] bereitgestellt.

3.2.3.3 Lesen von der SD-Karte

Das Lesen aus einer auf der SD-Karte vorhandenen Datei wird mit Hilfe des folgenden Quelltextes ermöglicht.

```
if (fp != NULL)
{
	char c = fgetc(fp);
	if (c == 'T')
	{
		//erwartetes Zeichen gelesen
	}
	else
	{
		//nicht erwartetes Zeichen gelesen
	}
	fclose(fp);
}
else
{
	//Fehlerbehandlung keine Datei/SD-Karte
}
```

Nachdem der Zeiger auf die Datei mit `fopen` geöffnet wurde, kann mit Hilfe von verschiedenen Funktionen, beispielsweise der Funktion `fgetc`, aus der Datei gelesen werden. Die Funktion `fgetc` liest dabei ein Zeichen aus und gibt dieses zurück. Als Argument erwartet `fgetc` nur den Zeiger auf die Datei.

Die Beschreibung der Funktion `fgetc` wird auf der Webseite opengroup.org [41] bereitgestellt.

3.3 Programmierung

Im Folgenden wird aufgezeigt, mit welchen Werkzeugen dieses Projekt realisiert wurde. Weiterhin werden verwendete Bibliotheken vorgestellt und deren Modifikationen aufgezeigt.

Abschließend wird beschrieben, wie die verwendeten Hardwarekomponenten konfiguriert werden müssen.

3.3.1 Entwicklungsumgebung und Software Development Kit

Für dieses Projekt werden eine Entwicklungsumgebung und ein Software Development Kit (SDK) verwendet. Diese werden im Folgenden vorgestellt.

3.3.1.1 mbed Compiler

Der mbed Compiler ist eine Entwicklungsumgebung, die im Browser ausgeführt wird. Nachdem im Entwicklerbereich [42] ein kostenloses mbed-Konto erstellt wurde, kann der Compiler über die Webseite [43] aufgerufen werden.

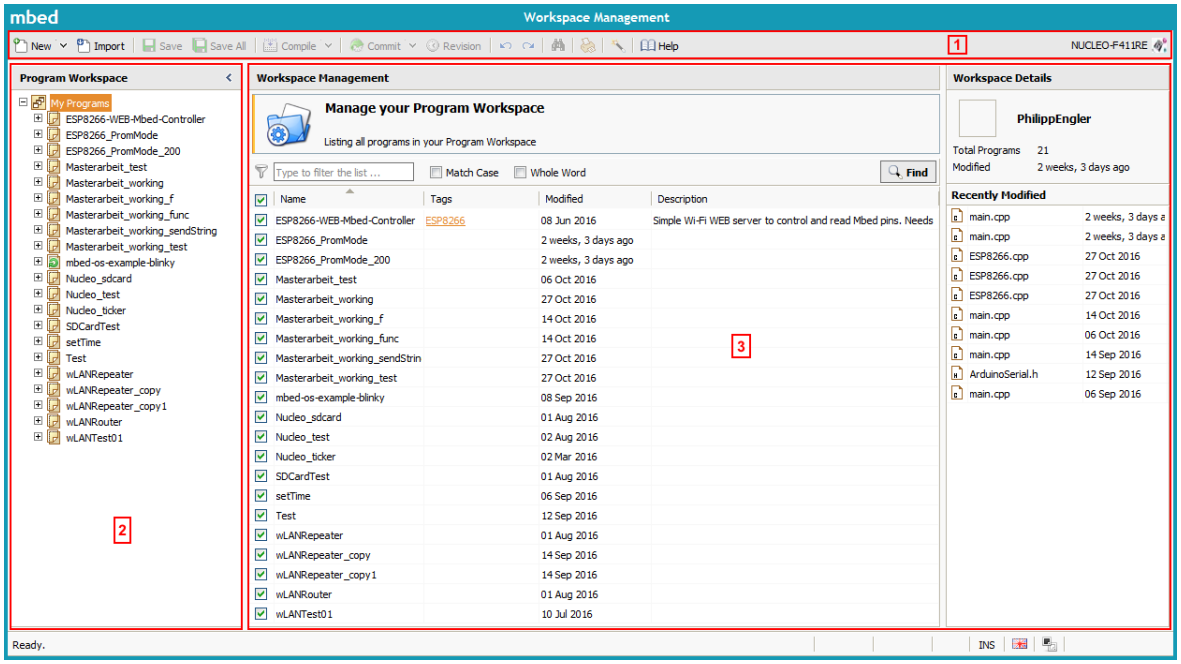

Abbildung 10: Übersicht über die Bereiche des mbed Compilers

Die Entwicklungsumgebung ist in verschiedene Bereiche unterteilt. Diese werden in der Abbildung 10 dargestellt. Der Bereich 1 beinhaltet die Menüleiste, über die verschiedene Befehle aufgerufen werden können. Eine Liste der unter dem aktuellen Benutzerkonto erstellten Programme befindet sich im Bereich 2. Im Bereich 3 befindet in Abhängigkeit der Auswahl im Bereich 2 entweder eine Liste der Programme, eine Liste der im Programm enthaltenen Dateien und Bibliotheken oder der Quellcode.

Bei der Erstellung eines neuen Programms mittels NEW | NEW PROGRAM wird zuerst eine Zielplattform festgelegt, anschließend kann entweder eine Vorlage ausgewählt werden oder ein leeres Programm erstellt werden.

Wurde ein leeres Programm erstellt, muss als erstes dem Programm eine neue Datei `main.cpp` hinzugefügt werden. Nach dem Öffnen der Datei kann diese bearbeitet werden.

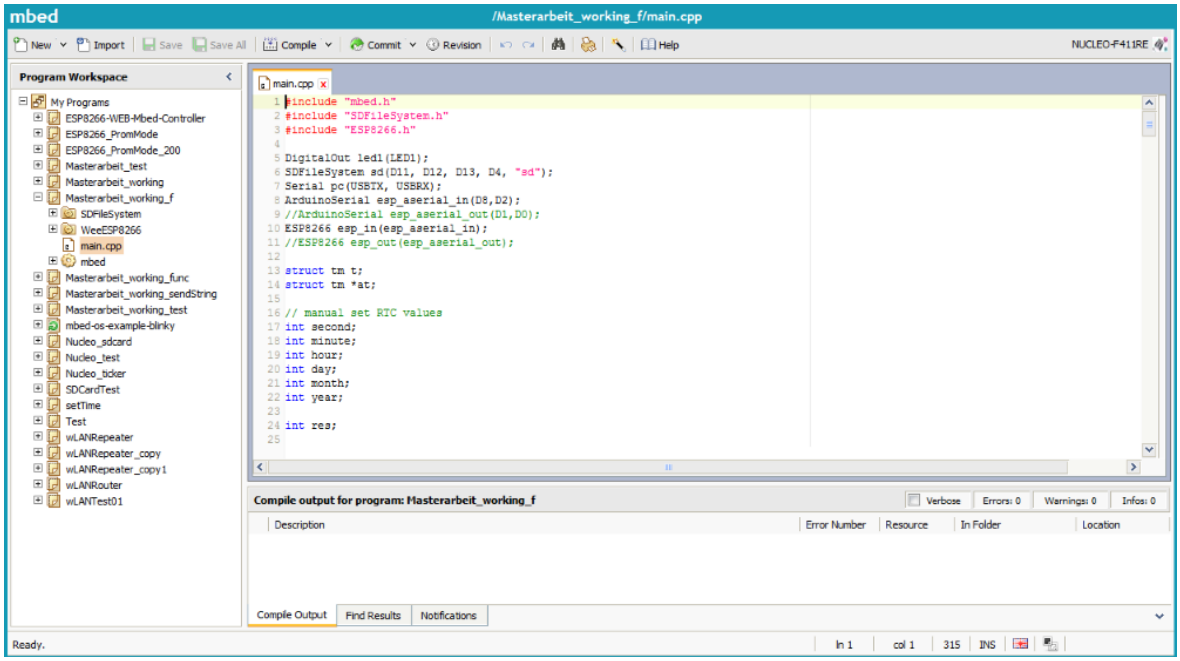

Abbildung 11: mbed Compiler im Editiermodus

Zu verwendende Bibliotheken können auf verschiedenen Wegen eingebunden werden.

Der schnellste Weg ist eine Einbindung mittels des `include`-Befehls, gefolgt von der Angabe einer Header-Datei.

```
#include "mbed.h"
```

Mit Hilfe des oben gezeigten Beispiels lässt sich die mbed-Bibliothek importieren. Die Bibliothek wird beim nächsten Durchlauf des Compilers zur Auswahl angeboten und anschließend importiert.

Eine weitere Möglichkeit ist der Import einer Bibliothek von mbed.org. Dazu wird in der Menüleiste auf IMPORT geklickt. Anschließend erscheint ein Dialog, mit dessen Hilfe nach Bibliotheken gesucht werden kann. Aus diesen Ergebnissen kann die entsprechende Bibliothek durch Anklicken importiert werden. Nach dem Import muss die Bibliothek noch mittels des `include`-Befehls in das Programm eingefügt werden.

Wurde das Programm fertig gestellt, kann es über den in der Menüleiste enthaltenen Menüpunkt COMPILE erstellt werden. Nach der erfolgreichen Erstellung des Programms wird die erstellte Version automatisch zum Herunterladen angeboten.

Treten beim Erstellen des Programms Fehler auf, werden diese unterhalb des Quelltextes in einer Liste angezeigt. Diese Fehler müssen behoben werden, damit das Programm erstellt werden kann.

Die erstellte `*.bin`-Datei wird heruntergeladen und anschließend auf den Wechselmedienspeicher des per USB angeschlossenen STM32 Nucleo kopiert. Unmittelbar nach dem Kopieren beginnt der STM32 Nucleo automatisch die Abarbeitung des Programms.

Dieser Entwicklungsumgebung fehlt ein Debugging-Werkzeug. Der einzige Weg, das Programm auf dem STM32 Nucleo zu debuggen, besteht darin, die Debugging-Ausgabe über den seriellen Port zu nutzen. Diese Funktion wird im Kapitel 3.3.4.1 beschrieben.

3.3.1.2 Espressif ESP8266 SDK

Mit Hilfe des von Espressif [44] bereitgestellten SDK lässt sich eine individuelle Firmware für den ESP8266 erstellen.

Zur Erzeugung der Firmware werden bestimmte Werkzeuge benötigt. Damit dies vereinfacht wird, stellt Espressif unter [45] eine virtuelle Maschine auf der Basis von Linux zur Verfügung, die mittels Oracles VirtualBox genutzt werden kann. Oracles VirualBox steht unter dem oben genannten Downloadlink ebenfalls zur Verfügung.

Im ESP8266 SDK Getting Started Guide [46] befindet sich eine Schritt-für-Schritt-Anleitung zur Erstellung der Firmware sowie zum Übertragen dieser auf den ESP8266.

Mit der Erstellung einer angepassten Version der Firmware für das Drahtlosnetzwerkmodul ESP8266 ist es möglich, einen eigenen AT-Befehl zu programmieren, der den Promiscous-Mode aktiviert.

3.3.2 Verwendete Bibliotheken

Für dieses Projekt wurden zwei fertige Bibliotheken verwendet. Diese werden im Folgenden kurz beschrieben.

3.3.2.1 Bibliothek ‚SDFileSystem‘

Die für das SD Card Shield V4 verwendete Bibliothek ‚SDFileSystem‘ [37] stammt von ARMmbed.

Diese Bibliothek ermöglicht es einerseits, dass mit der SD-Karte kommuniziert werden kann, andererseits stellt diese Bibliothek die nötigen Funktionen zur Verfügung, um auf das FAT-Dateisystem der SD-Karte zugreifen zu können.

Die Kommunikation zwischen dem STM32 Nucleo und dem SD Card Shield V4 erfolgt wie bereits beschrieben über das Serial Peripheral Interface.

In diesem Projekt werden zwei Funktionen aus der Bibliothek genutzt. Zum einen ist dies der nachfolgend angegebene Konstruktor.

```
SDFileSystem(PinName mosi, PinName miso, PinName sclk, PinName cs,
const char* name);
```

Dieser Konstruktor erzeugt das Objekt, mit dem anschließend eine Kommunikation mit der SD-Karte stattfinden kann.

Die zweite Funktion dient zur Initialisierung der SD-Karte.

```
disk_initialize();
```

Diese Funktion initialisiert unter anderem die Anzahl der Sektoren und die Blockgröße sowie die Datentransferrate zur Datenübertragung zwischen den STM32 Nucleo und den SD Card Shield V4.

Ein weiterer Zugriff auf die SD-Karte erfolgt anschließend ausschließlich über die Funktionen der Standard-C-Bibliothek `stdio`.

Bei der Verwendung des SD Card Shield V4 ist zu beachten, dass das Modul die LED `LD2` für die Anzeige von Schreibe- und Lesevorgängen benutzt. Dadurch ist die Verwendung der LED `LD2` für eigene Zwecke nicht mehr möglich.

3.3.2.2 Bibliothek ‚WeeESP8266'

Wie bereits im Kapitel 3.1.4 beschreiben, lässt sich das Drahtlosnetzwerk-Modul ESP8266 mittels AT-Befehle steuern.

Die in diesem Projekt verwendete Bibliothek WeeESP8266 [47] von ITEAD STUDIO verwendet ebenfalls diese AT-Befehle, stellt diese jedoch über Funktionsnamen zur Verfügung und ermöglicht es außerdem, den Datenverkehr über geöffnete TCP-Ports zu erfassen.

Da es sich bei dieser Bibliothek um einen Fork von der Arduino-Plattform handelt, werden einige Funktionen der Arduino-Plattform genutzt. Diese Funktionen sind in der Bibliothek WeeESP8266 als Bibliothek ArduinoAPI [48] enthalten, die ebenfalls von ITEAD STUDIO entwickelt wurde.

Im Folgenden werden die aus dieser Bibliothek in diesem Projekt verwendeten Funktionen charakterisiert.

Konstruktor

Die Bibliothek enthält den folgenden Konstruktor.

```
ESP8266(ArduinoSerial &uart);
```

Dieser erzeugt ein Objekt, mit dem auf das Drahtlosnetzwerk-Modul ESP8266 zugegriffen werden kann. Dem Konstruktor wird ein Objekt vom Typ `ArduinoSerial` übergeben. Um das `ArduinoSerial`-Objekt zu erstellen, müssen dem Konstruktor die beiden PIN-Bezeichnungen der beiden PINs, an denen der Transmit- und der Receive-PIN des Drahtlosnetzwerk-Moduls angeschlossen wurden, übergeben werden.

```
ArduinoSerial esp_aserial(D8,D2);
ESP8266 esp(esp_aserial);
```

Im Programm kann nun mittels des Objektnamens `esp` auf die Funktionen des Drahtlosnetzwerk-Moduls zugegriffen werden.

Test des Drahtlosnetzwerk-Moduls

Die Funktionsfähigkeit des Drahtlosnetzwerk-Modul ESP8266 kann mit der folgenden Funktion überprüft werden.

```
kick();
```

Dazu übergibt die Funktion den AT-Befehl `AT` an das Drahtlosnetzwerk-Modul und wertet die Antwort aus. Ist in dieser Antwort die Zeichenkette `OK` enthalten, ist die Überprüfung des Drahtlosnetzwerk-Moduls erfolgreich verlaufen und die Funktion gibt den Wert `true` zurück, anderenfalls wird der Wert `false` zurückgegeben.

Neustart des Drahtlosnetzwerk-Moduls

Ein Neustart des Drahtlosnetzwerk-Moduls kann mit dem folgenden Befehl veranlasst werden. Dabei ist zu beachten, dass das Drahtlosnetzwerk-Modul etwa drei Sekunden für die Ausführung benötigt. Dabei werden alle eventuell bestehenden Verbindungen unterbrochen.

```
restart();
```

Um den Neustart auszuführen, wird an das Drahtlosnetzwerk-Modul der AT-Befehl `AT+RST` gesendet.

Version des Drahtlosnetzwerk-Moduls

Die folgende Funktion ermittelt die aktuell verwendete Firmwareversion des Drahtlosnetzwerk-Moduls.

```
getVersion();
```

An das Drahtlosnetzwerk-Modul wird der AT-Befehl `AT+GMR` gesendet. Aus der Antwort wird der Versionsstring herausgefiltert und anschließend zurückgegeben.

Betriebsmodus festlegen

Das Drahtlosnetzwerk-Modul kann in verschiedenen Betriebsmodi betrieben werden. Es kann immer nur ein Modus ausgewählt werden. Das Setzen eines neuen Modus ersetzt den zuletzt gewählten Modus.

Jeder dieser Modi wird mit Hilfe eines AT-Befehls im Drahtlosnetzwerk-Modul gesetzt. Die jeweilige Funktion überprüft vorher, ob der Modus bereits gesetzt ist; ist dies nicht der Fall, wird er gesetzt und das Drahtlosnetzwerk-Modul neu gestartet.

Betriebsmodus „Station“

Im sogenannten Station-Modus verhält sich das Drahtlosnetzwerk-Modul wie ein Endgerät. Dieser Modus kann mit dem folgenden Befehl gesetzt werden.

```
setOprToStation();
```

An das Modul wird der Befehl `AT+CWMODE=1` übertragen.

Betriebsmodus „SoftAP“

Wird der Modus SoftAP gesetzt, verhält sich das Drahtlosnetzwerk-Modul als Drahtlosnetzwerk Access Point und ermöglicht somit, dass sich Endgeräte mit dem Modul verbinden können.

```
setOprToSoftAP();
```

Zur Ausführung überträgt diese Funktion den Befehl `AT+CWMODE=2` an das Drahtlosnetzwerk-Modul.

Betriebsmodus „Station and SoftAP“

Das Drahtlosnetzwerk-Modul kann auch gleichzeitig Endgerät und Drahtlosnetzwerk Access Point sein. Dazu muss die folgende Funktion ausgeführt werden.

```
setOprToStationSoftAP();
```

Dieser Modus wird mit dem Befehl `AT+CWMODE=3` gesetzt.

Je nach gewähltem Betriebsmodus können weitere Funktionen ausgeführt werden, etwa um eine Liste aller Access Points in der Umgebung abzurufen (`getAPList()`), sich mit

einem Access Point zu verbinden (`joinAP(ssid, pwd)`) oder diese Verbindung zu lösen (`leaveAP()`), um die IP-Adressen der verbundenen Geräte zu erhalten (`getJoinedDeviceIP()`) oder die IP-Adresse des Drahtlosnetzwerk-Moduls abzurufen (`getLocalIP()`). Jede der eben genannten Funktionen sendet an das Drahtlosnetzwerk-Modul entsprechende AT-Behle. Eine Übersicht über diese AT-Befehle befindet sich beispielsweise bei room-15 [49].

Konfiguration des SoftAP

Wird das Drahtlosnetzwerk-Modul im Access Point Modus betrieben, muss dieser Access Point konfiguriert werden. Die folgende Funktion zeigt, wie die Konfiguration erfolgt.

```
setSoftAPParam(ssid, pwd, chl, ecn);
```

Dazu werden der Funktion vier Argumente übergeben:

- `ssid` – SSID des aufzuspannenden Drahtlosnetzwerks
- `pwd` – das Passwort des aufzuspannenden Drahtlosnetzwerks
- `chl` – die Nummer des Kanals, auf dem das Modul senden soll
- `enc` – die Verschlüsselungsmethode

Dabei ist zu beachten, dass der Kanal durch eine Zahl zwischen `1` und `13` repräsentiert wird. Für die Verschlüsselungsmethode stehen die folgenden Möglichkeiten zur Auswahl:

- `0` – ohne Verschlüsselung
- `1` – WEP
- `2` – WPA (PSK)
- `3` – WPA2 (PSK)
- `4` – WPA (PSK) + WPA2 (PSK)

Die Konfiguration des Drahtlosnetzwerk-Moduls erfolgt mit dem AT-Befehl `AT+CWSAP=ssid,pwd,chl,ecn`. Dabei müssen die Platzhalter `ssid`, `pwd`, `chl` und `enc` durch die entsprechenden Werte ersetzt werden.

Anzahl der Verbindungsendpunkte

Mit Hilfe der nachfolgenden Funktionen kann festgelegt werden, ob nur ein Verbindungsendpunkt unterstützt wird oder ob es möglich sein soll, bis zu vier Verbindungsendpunkte zu erzeugen.

Wird die nachstehende Funktion aufgerufen, können bis zu vier Endpunkte erstellt werden.

```
enableMUX();
```

Dazu wird an das Drahtlosnetzwerk-Modul der AT-Befehl `AT+CIPMUX=1` gesendet. Wird in der Antwort des Drahtlosnetzwerk-Moduls `OK` gefunden, wird `true` zurückgegeben, anderenfalls wird `false` zurückgegeben.

Um festzulegen, dass nur ein Endpunkt erstellt werden kann, muss die folgende Funktion aufgerufen werden.

```
disableMUX();
```

In diesem Fall wird an das Drahtlosnetzwerk-Modul der AT-Befehl `AT+CIPMUX=0` gesendet. Auch in diesem Fall wird `true` zurückgegeben, wenn in der Antwort des Drahtlosnetzwerk-Moduls `OK` gefunden wurde, anderenfalls wird `false` zurückgegeben.

TCP-Server

Starten des TCP-Servers

Damit ein TCP-Server auf einem beliebigen Port erstellt werden kann, wird die folgende Funktion genutzt. Ein TCP-Server kann nur erstellt werden, wenn das Drahtlosnetzwerk-Modul mehrere Verbindungsendpunkte unterstützt, also nach einem Aufruf von `enableMUX()`.

```
startTCPServer(port);
```

Dabei enthält die Variable `port` eine beliebige Portnummer, beispielsweise `80`. Die Funktion sendet den AT-Befehl `AT+CIPSERVER=1,port` an das Drahtlosnetzwerk-Modul und wertet die Antwort aus. Im Erfolgsfall wird `true`, ansonsten `false` zurückgegeben.

Beenden des TCP-Servers

Ein aktiver TCP-Server kann mit Hilfe des nachstehenden Befehls beendet werden.

```
stopTCPServer();
```

Damit dem Drahtlosnetzwerk-Modul die Änderung bekannt gemacht werden kann, wird der AT-Befehl `AT+CIPSERVER=0` an das Drahtlosnetzwerk-Modul gesendet. Je nach Antwort des Moduls wird entweder `true` im Erfolgsfall oder `false` im Fehlerfall zurückgegeben.

Daten empfangen

Damit die Daten, die während einer TCP-Verbindung zwischen einem Endgerät und dem Drahtlosnetzwerk-Modul ausgetauscht werden, ausgewertet werden können, kann eine der folgenden Funktionen genutzt werden.

```
recv(buffer, buffer_size, timeout);
recv(mux_id, buffer, buffer_size, timeout);
```

Der Unterschied zwischen beiden Methoden besteht darin, dass die Funktion `recv(buffer, buffer_size, timeout)` sämtliche empfangenen Daten zurückgibt, im Gegensatz zur Funktion `recv(mux_id, buffer, buffer_size, timeout)`, die nur die Daten mit der entsprechenden ID zurückgibt.

Daten senden

Damit Daten über eine TCP-Verbindung gesendet werden können, kann eine der folgenden Funktionen genutzt werden.

```
send(buffer, len);
send(mux_id, buffer, len)
```

Der Unterschied zwischen beiden Methoden besteht darin, dass die Funktion `send(buffer, len)` Daten im Modus für einen Verbindungsendpunkt versendet, im Gegensatz zur Funktion `send(mux_id, buffer, len)`, die Daten über die Verbindung mit der entsprechenden ID sendet.

Es existieren in der Bibliothek WeeESP8266 weitere Funktionen zum Steuern und zur Nutzung des Drahtlosnetzwerk-Moduls, die in diesem Projekt allerdings nicht verwendet werden. Eine Beschreibung aller Funktionen befindet sich in der Datei `ESP8266.h`.

3.3.3 Anpassung der Bibliotheken

Für die Nutzung der fertigen Bibliotheken in diesem Projekt mussten Anpassungen vorgenommen werden, die im Folgenden vorgestellt werden.

3.3.3.1 Bibliothek SDFileSystem

Die Bibliothek zur Nutzung des SD Card Shield V4 wurde ohne weitere Anpassungen übernommen.

3.3.3.2 Bibliothek WeeESP8266

In dieser Bibliothek wurde die Übertragungsgeschwindigkeit der Daten zwischen dem STM32 Nucleo und den Drahtlosnetzwerk-Modulen verändert sowie je zwei öffentliche und private Funktionen hinzugefügt. Die beiden neuen öffentlichen Funktionen ermöglichen es, empfangene Daten auf die SD-Karte auszuleiten. Eine der beiden öffentlichen Funktionen kann zusätzlich die empfangenen Daten über das zweite Drahtlosnetzwerk-Modul weiterleiten.

Übertragungsgeschwindigkeit

Das in diesem Projekt verwendete Drahtlosnetzwerk-Modul unterstützt eine Übertragungsgeschwindigkeit von 115200 Baud. Die Bibliothek initialisiert das Drahtlosnetzwerk-Modul jedoch mit 9600 Baud. Somit muss in der Datei `ESP8266.cpp` in der Funktion `ESP8266::ESP8266(ArduinoSerial &uart): m_puart(&uart)` die Zeile

```
m_puart->begin(9600);
```

durch die Zeile

```
m_puart->begin(115200);
```

ersetzt werden.

Funktion zum Empfangen der Daten und Ausleiten auf SD-Karte

Die Bibliothek WeeESP8266 stellt zwei Funktionen zum Empfangen und Verarbeiten von Daten aus TCP-Verbindungen bereit. Beide Funktionen nutzen die gleiche private Funktion, um die Daten auszugeben.

Damit die Daten ausgeleitet werden können, wurde die neue öffentliche Methode `recv(timeout, fp)` erstellt, der eine Zeitspanne (`timeout`) sowie ein Zeiger auf eine Datei (`fp`) übergeben wird. Die Zeitspanne dient dazu, das Empfangen der Daten nach der angegebenen Zeit in Millisekunden automatisch zu beenden.

Der Quelltext der Methode ist nachfolgend dargestellt.

```
uint32_t ESP8266::recv(uint32_t timeout, FILE *fp)
{
      return recvPkg(timeout, fp);
}
```

Diese Methode ruft die ebenfalls neu erstellte private Methode `recvPkg(timeout, fp)` auf. Der Methode werden ebenfalls die Argumente Zeitspanne (`timeout`) und Zeiger auf eine Datei (`fp`) übergeben.

Das Empfangen der Daten findet in der `while`-Schleife statt, die nicht durchlaufen wird, wenn die übergebene Zeitspanne abgelaufen ist. Die Zeichen werden aus der UART-Schnittstelle ausgelesen, zwischengespeichert und anschließend an einen String angehängt. Nachdem keine Daten mehr bereitstehen, wird die Zeichenkette in die Datei geschrieben und die `while`-Schleife wird – wenn die Zeitspanne noch nicht abgelaufen ist – erneut durchlaufen.

Da die Bedingung zur Ausführung der `while`-Schleife am Anfang geprüft wird, wird während der Ausführung der Schleife nicht erkannt, ob die Zeitspanne bereits abgelaufen ist. Dies hat den Vorteil, dass der Schleifendurchlauf immer beendet wird, gleichzeitig kann es aber zu einer geringen Überschreitung der Zeitspanne kommen. Allerdings ist dadurch sichergestellt, dass die empfangenen Daten komplett auf die SD-Karte geschrieben werden.

Im Folgenden ist der Quelltext der Methode abgebildet.

```
uint32_t ESP8266::recvPkg(uint32_t timeout, FILE *fp)
{
	String data;
	char a;

	unsigned long start = millis();
	while (millis() - start < timeout)
	{
		while(m_puart->available() > 0)
		{
			a = m_puart->readChr();
			data += a;
		}

		fp = fopen("/sd/data.txt", "a");
		if (fp != NULL)
		{
			fprintf(fp, "%s", data.c_str());
			fclose(fp);
		}
		else
		{
			return 1;
		}
	}
	return 0;
}
```

Funktion zum Empfangen der Daten und Ausleiten auf SD-Karte und Weiterleiten der Daten über ein zweites Drahtlosnetzwerk-Modul

Die eben vorgestellte Methode erlaubt es nicht, die empfangen Daten über ein zweites Drahtlosnetzwerk-Modul weiterzuleiten. Aus diesem Grund wurde die öffentliche Methode `recv(timeout, esp_out, fp)` erstellt, der neben der Zeitspanne (`timeout`) und dem Zeiger auf eine Datei (`fp`) das Objekt des zweiten Drahtlosnetzwerk-Moduls (`esp_out`) übergeben wird.

Im Folgenden wird der Quelltext der Methode dargestellt.

```
uint32_t ESP8266::recv(uint32_t timeout, ESP8266 esp_out, FILE *fp)
{
      return recvPkg(timeout, esp_out, fp);
}
```

Diese Methode ruft die ebenfalls neu erstellte private Methode `recvPkg(timeout, esp_out, fp)` auf. Der Methode werden ebenfalls die Argumente Zeitspanne (`timeout`), das Objekt des zweiten Drahtlosnetzwerk-Moduls (`esp_out`) und ein Zeiger auf eine Datei (`fp`) übergeben.

Der Ablauf der Methode entspricht im Grunde dem der Methode `recvPkg(timeout, fp)`. Das Empfangen der Daten findet in der `while`-Schleife statt, die nicht durchlaufen wird, wenn die übergebene Zeitspanne abgelaufen ist. Die Zeichen werden aus der UART-Schnittstelle ausgelesen, zwischengespeichert und anschließend an einen String angehängt. Nachdem keine Daten mehr bereitstehen, werden die Daten über das zweite Drahtlosnetzwerk-Modul gesendet. Danach wird die Zeichenkette zusammen mit einem Zeitstempel in die Datei geschrieben und anschließend wird die `while`-Schleife – wenn die Zeitspanne noch nicht abgelaufen ist – erneut durchlaufen.

Wie bereits ausgeführt, wird die Bedingung zur Ausführung der `while`-Schleife am Anfang geprüft. Somit wird der Schleifendurchlauf immer beendet, allerdings kann es zu einer Überschreitung der vorgegebenen Zeitdauer kommen. Dadurch ist sichergestellt, dass die empfangenen Daten komplett weitergeleitet und auf die SD-Karte geschrieben werden.

Im Folgenden ist der Quelltext der Methode abgebildet.

```
uint32_t ESP8266::recvPkg(uint32_t timeout, ESP8266 esp_out, FILE *fp)
{
	String data;
	char a;

	unsigned long start = millis();
	while (millis() - start < timeout)
	{
		while(m_puart->available() > 0)
		{
			a = m_puart->readChr();
			data += a;
		}
		bool resValueBool = esp_out.send((uint8_t*)data,
				sizeof((uint8_t*)data));
		int cmp=strcmp(data.c_str(), "\r\n");
		if(cmp!=0)
		{
			fp = fopen("/sd/rec_data.txt", "a");
			if (fp != NULL)
			{
				struct tm *actt;
				time_t actTime;
				time(&actTime);
				actt=localtime(&actTime);
				fprintf(fp, "\r\n\r\n---\r\n%s\r\n\r\n",
					asctime(actt));
				fprintf(fp, "%s", data.c_str());
				fclose(fp);
			}
			else
			{
				return 1;
			}
		}
	}
	return 0;
}
```

3.3.4 Grundlegendes zur Verwendung des STM32 Nucleo

In diesem Kapitel werden die grundlegende Einrichtung und Konfiguration des STM32 Nucleo sowie die Verwendung des Drahtlosnetzwerk-Moduls ESP8266 und des SD Card Shield V4 vorgestellt.

3.3.4.1 Ausgabe von Statusmeldungen des STM32 Nucleo

Damit Statusmeldungen vom STM32 Nucleo auf einem Computer angezeigt werden können, muss eine serielle Verbindung hergestellt werden. Dazu werden die beiden Ports `USBTX` und `USBRX` genutzt. Diese STLink Debug-Verbindung setzt den Anschluss Serial2 (PINs `D0` und `D1`) auch dann außer Betrieb, wenn diese Verbindung nicht genutzt wird. Im Folgenden wird dargestellt, wie die serielle Verbindung initialisiert wird.

```
Serial pc(USBTX, USBRX);
```

Danach kann innerhalb der Anwendung eine Statusmeldung wie folgt ausgegeben werden.

```
pc.printf("Setting up environment...\r\n");
```

Zur Nutzung muss ein spezieller Treiber von der Webseite [50] heruntergeladen werden. Nachdem der entsprechende Treiber installiert wurde, werden die Meldungen über einen COM-Port empfangen. Um diese Nachrichten anzeigen zu können, kann beispielsweise das Programm PuTTY [51] genutzt werden.

3.3.4.2 Einbindung des Drahtlosnetzwerk-Moduls ESP8266

Die Bibliothek WeeESP8266 wird mit dem folgenden Befehl in das Programm eingebunden.

```
#include "ESP8266.h"
```

Anschließend muss zuerst für jedes Drahtlosnetzwerk-Modul ein `ArduinoSerial`-Objekt erzeugt werden. Diesem `ArduinoSerial`-Objekt sind die genutzten Anschlüsse zu übergeben.

Danach kann für jedes Drahtlosnetzwerk-Modul ein `ESP8266`-Objekt erzeugt werden, dem das entsprechende `ArduinoSerial`-Objekt übergeben wird.

```
ArduinoSerial espaserial_in(D8,D2);
ArduinoSerial espaserial_out(PA_11,PA_12);

ESP8266 esp_in(espaserial_in);
ESP8266 esp_out(espaserial_out);
```

Im Programm kann nun über das entsprechende ESP8266-Objekt auf die Methoden zur Steuerung des Drahtlosnetzwerk-Moduls zugegriffen werden.

Der folgende Quelltextauszug verdeutlicht die Nutzung der Bibliothek am Beispiel der Testmethode `kick()` für das Drahtlosnetzwerk-Modul sowie die Auswertung des Ergebnisses der Funktion.

```
bool resValueBool = esp_out.kick();
if(resValueBool == 0)
{
      pc.printf(" failed.\r\n");
      wlanfail();
}
else
{
      pc.printf(" success.\r\n");
}
```

3.3.4.3 Einbindung des SD Card Shield V4

Folgender Befehl wird zur Einbindung des SD Card Shield V4 in das Programm verwendet.

```
#include "SDFileSystem.h"
```

Damit auf die SD-Karte zugegriffen werden kann, muss das `SDFileSystem`-Objekt erstellt werden. Dies geschieht mit dem folgenden Befehl.

```
SDFileSystem sd(D11, D12, D13, D4, "sd");
```

Nachdem das `SDFileSystem`-Objekt erstellt wurde, muss dieses initialisiert werden. Diese Initialisierung erfolgt mit dem Befehl `disk_initialize()`, dessen Nutzung im folgenden Quelltextauszug gezeigt wird. Das Ergebnis der Funktion enthält den Typ der SD-Karte.

```
pc.printf("Initialize SD card...");

res = sd.disk_initialize();

if(res==0)
{
      pc.printf(" success. Type of SD Card: %d.\r\n", res);
}
else
{
      pc.printf(" failed.\r\n");
      sdfail();
}
```

Ein Zugriff auf die SD-Karte erfolgt über die Funktionen der Standard-C-Bibliothek `stdio`. Am Beispiel des Befehls zum Öffnen einer Datei wird die Nutzung zum Zugriff auf die SD-Karte gezeigt.

```
FILE *fp = fopen("/sd/config/date.txt", "r");
if (fp != NULL)
{
      pc.printf(" done.\r\n", ssid);
}
else
{
      pc.printf(" failed.\r\n");
}
```

3.3.4.4 Setzen der Uhrzeit

Der STM32 Nucleo verfügt nicht über eine batteriegepufferte Echtzeituhr. Eine solche Echtzeituhr, die auch Real Time Clock oder RTC genannt wird, stellt durch die Pufferung die korrekte Uhrzeit auch nach einem Ausfall der Betriebsspannung zur Verfügung.

Damit in diesem Projekt die Uhrzeit gesetzt werden kann, fragt der STM32 Nucleo nach der Initialisierung die Uhrzeit von der SD-Karte ab. Dazu liegt im Verzeichnis `config` eine Datei `date.txt`, die beim Starten des STM32 Nucleo eingelesen wird. Diese enthält das Datum und die Uhrzeit, die nach dem Start gesetzt werden soll. Der Inhalt der Datei ist eine Zeichenkette mit dem Format `yyyy mm dd hh mm ss`. Im Folgenden wird ein Beispiel dargestellt.

```
2016 11 03 07 51 53
```

Nach dem Einlesen der Zeichenkette werden die Variablen entsprechend gesetzt und eine Funktion aufgerufen, die das Datum und die Uhrzeit des STM32 Nucleo auf die eingegebenen Werte einstellt.

```
pc.printf("Reading date/time from SD card...");
FILE *fp = fopen("/sd/config/date.txt", "r");
if (fp != NULL)
{
      fscanf(fp, "%d %d %d %d %d %d", &year, &month, &day, &hour,
                        &minute, &second);
      fclose(fp);
      pc.printf(" readed: Year: %d; Month: %d, Day: %d, Hour: %d,
                        Minute: %d, Second: %d\r\n", year, month, day,
                        hour, minute, second);
      pc.printf("Setting date/time to readed values...");
      setRTC();
      pc.printf(" done.\r\n", ssid);
}
else
{
      pc.printf(" failed.\r\n");
}
```

In der Funktion zum Stellen der Uhrzeit des STM32 Nucleo muss beachtet werden, dass das `time`-Objekt der Programmiersprache C zwei Besonderheiten enthält:

- Die Zählung des Monats beginnt bei `0`. Somit muss vom eingelesenen Wert eins subtrahiert werden.
- Die Jahreszahl wird als Wert ab `1900` dargestellt. Somit muss von der Jahreszahl der Wert `1900` subtrahiert werden.

Nachdem diese Korrekturen vorgenommen wurden, kann das `time`-Objekt erstellt und die Uhrzeit des STM32 Nucleo gesetzt werden.

```
void setRTC()
{
      t.tm_sec = (second);
      t.tm_min = (minute);
      t.tm_hour = (hour);
      t.tm_mday = (day);
      t.tm_mon = (month-1);
      t.tm_year = ((year)-1900);
      set_time(mktime(&t));
}
```

Das vom STM32 Nucleo genutzte Datum und die Uhrzeit können mit der folgenden Funktion ermittelt werden.

```
void getTime()
{
      time_t aTime;
      time(&aTime);
      at=localtime(&aTime);
      pc.printf("Getting current system date/time: ");
      pc.printf(" %s\r\n", asctime(at));
}
```

3.3.4.5 Konfiguration des Drahtlosnetzwerk-Moduls

Damit die Drahtlosnetzwerk-Module wie gewünscht arbeiten, müssen diese konfiguriert werden. Da beide Drahtlosnetzwerk-Module in unterschiedlichen Betriebsmodi arbeiten, erfolgt eine individuelle Konfiguration der beiden Drahtlosnetzwerk-Module.

Vor einer Konfiguration sollte überprüft werden, ob die Drahtlosnetzwerk-Module ansprechbar sind. Dies kann mit der Funktion `kick()` erfolgen.

```
pc.printf("Testing outbound WLAN Card...");
esValueBool = esp_out.kick();
if(resValueBool == 0)
{
      pc.printf(" failed.\r\n");
      wlanfail();
}
else
{
      pc.printf(" success.\r\n");
}

pc.printf("Testing inbound WLAN Card...");
resValueBool = esp_in.kick();
if(resValueBool == 0)
{
      pc.printf(" failed.\r\n");
      wlanfail();
}
else
{
      pc.printf(" success.\r\n");
}
```

Nach dem Starten oder dem Neustart des STM32 Nucleo sollten die Drahtlosnetzwerk-Module ebenfalls neugestartet werden, damit sich diese in einem definierten Zustand befinden.

```
pc.printf("Restarting outbound WLAN Card...");
resValueBool = esp_out.restart();
if(resValueBool == 0)
{
      pc.printf(" failed.\r\n");
      wlanfail();
}
else
{
      pc.printf(" success.\r\n");
}

pc.printf("Restarting inbound WLAN Card...");
resValueBool = esp_out.restart();
if(resValueBool == 0)
{
      pc.printf(" failed.\r\n");
      wlanfail();
}
else
{
      pc.printf(" success.\r\n");
}
```

Zur Information kann die verwendete Firmwareversion des jeweiligen Drahtlosnetzwerk-Moduls ausgegeben werden.

```
pc.printf("Get version of outbound WLAN Card...");
resValueString = esp_out.getVersion();
pc.printf(" Version: %s\r\n", resValueString.c_str());

pc.printf("Get version of inbound WLAN Card...");
resValueString = esp_in.getVersion();
pc.printf(" Version: %s\r\n", resValueString.c_str());
```

Das Drahtlosnetzwerk-Modul, das die Daten an einen anderen Access Point weiterleitet, wird im sogenannten Station-Mode betrieben. Dazu wird das Drahtlosnetzwerk-Modul wie folgt konfiguriert.

```
pc.printf("Configuring outbound WLAN Card...\r\n");

pc.printf("Set outbound WLAN Card to station mode...");
resValueBool = esp_out.setOprToStation();
if(resValueBool == 0)
{
      pc.printf(" failed.\r\n");
      wlanfail();
}
else
{
      pc.printf(" success.\r\n");
}
```

Das andere Drahtlosnetzwerk-Modul wird im Access Point Modus genutzt. Für diesen Modus müssen mehrere Einstellungen vorgenommen werden.

Als erstes muss das Drahtlosnetzwerk-Modul in den Access Point Mode geschaltet werden. Dies geschieht mit folgendem Befehl.

```
resValueBool = esp_in.setOprToSoftAP();
if(resValueBool == 0)
{
      pc.printf(" failed.\r\n");
      wlanfail();
}
else
{
      pc.printf(" success.\r\n");
}
```

Anschließend muss der Modus für mehrere Verbindungen aktiviert werden.

```
resValueBool = esp_in.enableMUX();
if(resValueBool == 0)
{
      pc.printf(" failed.\r\n");
      wlanfail();
}
else
{
      pc.printf(" success.\r\n");
}
```

Danach muss die Access Point konfiguriert werden. Dazu werden als Argumente die SSID (`ssid`), das Passwort (im nachfolgenden Beispiel ohne Passwort), der Funkkanal (`7`) und der Verschlüsselungsmodus (`0` – keine Verschlüsselung) übergeben.

```
resValueBool = esp_in.setSoftAPParam(ssid, "", 7, 0);
if(resValueBool == 0)
{
      pc.printf(" failed.\r\n");
      wlanfail();
}
else
{
      pc.printf(" success.\r\n");
}
```

4 Diskussion

In diesem Kapitel findet eine Betrachtung des Ergebnisses in Hinblick auf den Datendurchsatz statt und es werden weitere Maßnahmen aufgezeigt, die für den geplanten Einsatz dieses Moduls von Bedeutung sind.

4.1 Vergleich

In diesem Projekt stellte sich heraus, dass die Menge der zu verarbeitenden Daten sehr hoch sein kann und es somit zu Problemen beim Empfang, bei der Verarbeitung und der Weiterleitung der Daten kommen kann.

Aus diesem Grund werden im Folgenden die von der verwendeten Hardware zu Verfügung gestellten Schnittstellen in Bezug auf die Datenübertragungsgeschwindigkeiten untersucht und die Verwendbarkeit in diesem Projekt bewertet.

Da die Daten über das Drahtlosnetzwerk-Modul empfangen werden, wird in einer ersten Betrachtung versucht, eine reelle Übertragungsgeschwindigkeit für Drahtlosnetzwerke nach dem Standard IEEE 802.11g im 2,4 Gigahertz Frequenzband zu ermitteln. Diese Übertragungsgeschwindigkeit wird als Grundlage für die Betrachtung der anderen Übertragungswege genutzt.

Die vom Drahtlosnetzwerk-Modul empfangenen Daten müssen anschließend dem STM32 Nucleo zur Verfügung gestellt werden. Für diesen Vorgang stehen zwei unterschiedliche Übertragungswege zur Verfügung.

Zunächst erfolgt die Betrachtung der in diesem Projekt verwendeten UART-Schnittstelle. Es wird die grundlegende Funktionsweise beschrieben sowie eine reell zu erzielende Übertragungsgeschwindigkeit bestimmt. Danach wird eine Einschätzung über die Eignung der UART-Schnittstelle für dieses Projekt vorgenommen.

Anschließend erfolgt die Betrachtung des alternativen Übertragungswegs über das Serial Peripheral Interface (SPI). Diese Schnittstelle wird für die Kommunikation mit dem SD Card Shield V4 verwendet und kann ebenfalls für die Kommunikation mit dem Drahtlosnetzwerk-Modul genutzt werden. Für die Nutzung mit dem Drahtlosnetzwerk-Modul sind jedoch Änderungen am Drahtlosnetzwerk-Modul nötig. Diese werden vorgestellt. Anschließend wird die grundlegende Funktionsweise beschrieben und die reell zu erzielende Übertragungsgeschwindigkeit bestimmt.

Jeder der im Folgenden betrachteten Übertragungswege wird in Abhängigkeit von der in diesem Projekt genutzten Hardware mit den zur Verfügung stehenden Rahmenbedingungen bewertet und die Nutzung für dieses Projekt bestimmt.

4.1.1 Datenübertragungsgeschwindigkeit im Wireless LAN

Das verwendete Drahtlosnetzwerk-Modul ESP8266 unterstützt Drahtlosnetzwerke im Frequenzband von 2,4 Gigahertz mit einer maximalen Geschwindigkeit von 54 Megabit je Sekunde. Wie bereits in Kapitel 1.3 erwähnt handelt es sich hierbei um einen theoretischen Wert. Dieser wird in der Praxis nicht erreichbar sein.

Die folgende Abschätzung soll ermitteln, welche Geschwindigkeit in einem IEEE 802.11g Drahtlosnetzwerk erzielt werden kann. Im Kapitel 8.6 des Buches [52] wird diese Betrachtung für ein IEEE 802.11b Drahtlosnetzwerk durchgeführt. Allerdings unterscheiden sich die übertragenen Datenframes zwischen den beiden Standards IEEE 802.11b und IEEE 802.11g. Im Kapitel 3.6.1 des Buches [52] findet sich die Beschreibung der Frames für die IEEE 802.11g Drahtlosnetzwerke.

In dieser Abschätzung wird die Übertragung von 1460 Byte über eine TCP-Verbindung betrachtet. Gemäß dem RFC 843 der Network Working Group of the IETF (Netzwerk-Arbeitsgruppe der Internet Engineering Task Force) [53] beträgt die maximale Größe des Datenfeldes 1500 Byte.

Werden mehr als 1500 Byte übertragen, müssen weitere Pakete gesendet werden, die ebenfalls jeweils 1500 Byte ausnehmen können. Diese Pakete sind genauso aufgebaut, wie es in der nachfolgenden Berechnung vorgestellt wird. Falls weniger als 1500 Byte gesendet werden, ist das Datenfeld nicht vollständig gefüllt.

Werden nicht 1500 Byte oder ein vielfaches davon übertragen, führt dies zu einem nicht vollständig gefüllten Datenfeld. Insbesondere bei der Übertragung von deutlich weniger als 1500 Byte führt dies zu einer geringeren Datenübertragungsgeschwindigkeit. Bedingt durch den gleichbleibenden Aufbau des Paketes sinkt die Datenübertragungsrate in diesem Falle deutlich.

Die Übertragung wird mit dem Senden der PLCP-Präambel eingeleitet. Diese kann entweder eine Größe von 72 Bit oder 144 Bit besitzen. Im Folgenden wird von einer kurzen Präambel (72 Bit) ausgegangen. Diese Präambel wird mit einer Geschwindigkeit von einem Megabit je Sekunde übertragen. Somit kann die Übertragungsdauer berechnet werden.

$$t_{Präambel} = \frac{\#Bit}{v_{\text{Übertragung}}} = \frac{72Bit}{1MBit/s} = \underline{72\mu s}$$

Nach der Übertragung der PLCP-Präambel erfolgt die Übertragung des PLCP-Headers. Dieser besitzt eine Größe von 48 Bit und wird mit einer Geschwindigkeit von zwei Megabit je Sekunde übertragen. Nachfolgend wird die Übertragungsdauer berechnet.

$$t_{Header} = \frac{\#Bit}{v_{\text{Übertragung}}} = \frac{48Bit}{2MBit/s} = \underline{24\mu s}$$

Im Anschluss an die Übertragung des PLCP-Headers erfolgt eine Trainingssequenz, die eine Dauer von acht Mikrosekunden in Anspruch nimmt.

$$t_{Training} = \underline{8\mu s}$$

Wurde das Training abgeschlossen, werden ein Guard-Intervall und ein Signalfeld übertragen, deren Übertragung vier Mikrosekunden dauert.

$$t_{Guard} = \underline{4\mu s}$$

Als nächstes werden der MAC-Header und die Frame Check Sequence (FCS) übertragen. Dazu werden 34 Byte Daten gesendet, die mit sechs Megabit je Sekunde übertragen werden.

$$t_{MACHeader} = \frac{\#Bit}{v_{\text{Übertragung}}} = \frac{34Byte \cdot 8}{6MBit/s} = \frac{272Bit}{6MBit/s} = \underline{45{,}33\mu s}$$

Nun erfolgt die Übertragung der Logical Link Control (LLC), des Subnetwork Access Protocol (SNAP) und der TCP/IP-Header. Diese Übertragung besitzt eine Größe von 48 Byte und wird mit einer Geschwindigkeit von 54 Megabit je Sekunde übertragen.

$$t_{TCPHeader} = \frac{\#Bit}{v_{\text{Übertragung}}} = \frac{48Byte \cdot 8}{54MBit/s} = \frac{384Bit}{54MBit/s} = \underline{7{,}11\mu s}$$

Danach erfolgt die Übertragung der eigentlichen Daten. Im Folgenden sollen 1460 Byte mit der maximal möglichen Geschwindigkeit von 54 Megabit je Sekunde übertragen werden. Die Übertragungsdauer lässt sich wie folgt bestimmen.

$$t_{Daten} = \frac{\#Bit}{v_{\text{Übertragung}}} = \frac{1460Byte \cdot 8}{54MBit/s} = \frac{11680Bit}{54MBit/s} = \underline{216{,}30\mu s}$$

Ist die Übertragung der Daten abgeschlossen, erfolgt eine Ruhezeit mit einer Länge von sechs Mikrosekunden.

$$t_{Ruhezeit} = \underline{6\mu s}$$

Nun sind die Daten übertragen worden. Die bisher benötigte Zeit kann wie folgt bestimmt werden.

$$t_{\text{Ü}Daten} = t_{Präambel} + t_{Header} + t_{Training} + t_{Guard} + t_{MACHeader} + t_{TCPHeader} + t_{Daten} + t_{Ruhezeit}$$

$$= 72\mu s + 24\mu s + 8\mu s + 4\mu s + 45{,}33\mu s + 7{,}11\mu s + 216{,}30\mu s + 6\mu s$$

$$= \underline{382{,}74\mu s}$$

Die während dieser Zeit übertragenen Bit werden wie folgt bestimmt.

$$\#Bit_{\text{Ü}Daten} = \#Bit_{Präambel} + \#Bit_{Header} + \#Bit_{MACHeader} + \#Bit_{TCPHeader} + \#Bit_{Daten}$$

$$= 72Bit + 48Bit + 272Bit + 384Bit + 11680Bit = \underline{12456Bit}$$

Zur Bestätigung des Empfangs erfolgt die Übertragung einer Bestätigung auf MAC-Ebene. Dabei handelt es sich um einen sogenannten ACK-Frame. Der ACK-Frame selbst ist 14 Byte groß, muss allerdings in einen Frame eingebunden werden. Die Übertragungsgeschwindigkeit des ACK-Frames beträgt 6 Megabit je Sekunde und die Übertragungsdauer kann wie folgt berechnet werden.

$$t_{ACKFrame} = t_{Präambel} + t_{Header} + t_{Training} + t_{Guard} + \frac{\#Bit}{v_{\text{Ü}bertragung}} + t_{Ruhezeit}$$

$$= 72\mu s + 24\mu s + 8\mu s + 4\mu s + \frac{14Byte \cdot 8}{6MBit/s} + 6\mu s$$

$$= 108\mu s + \frac{112Bit}{6MBit/s} + 6\mu s = 108\mu s + 18{,}67\mu s + 6\mu s$$

$$= \underline{132{,}67\mu s}$$

Eine weitere Bestätigung wird vom Empfänger auf der TCP-Ebene versendet. Die dafür notwendige Übertragungsdauer findet sich im Folgenden.

$$t_{TCPACK} = t_{Präambel} + t_{Header} + t_{Training} + t_{Guard} + t_{MACHeader} + t_{TCPHeader} + t_{Ruhezeit}$$

$$= 72\mu s + 24\mu s + 8\mu s + 4\mu s + 45{,}33\mu s + 7{,}11\mu s + 6\mu s$$

$$= \underline{166{,}44\mu s}$$

Auf der MAC-Ebene findet eine erneute Bestätigung des TCP-ACKs statt. Damit wird erneut die Übertragungsdauer für das ACK-Frame benötigt.

$$t_{ACKFrame} = \underline{132{,}67\mu s}$$

Für die Bestätigung nach dem Empfang der Daten wird die Übertragungsdauer wie folgt bestimmt.

$$t_{\text{Ü}Best} = t_{ACKFrame} + t_{TCPACK} + t_{ACKFrame} = 132{,}67\mu s + 166{,}44\mu s + 132{,}67\mu s$$

$$= \underline{431{,}48\mu s}$$

Die übermittelte Datenmenge für die Bestätigungssequenz lässt sich wie folgt ermitteln. Zunächst erfolgen dazu einige Nebenrechnungen.

$$\#Bit_{ACKFrame} = \#Bit_{Präambel} + \#Bit_{Header} + 122Bit$$

$$= 72Bit + 48Bit + 122Bit = \underline{242Bit}$$

$$\#Bit_{TCPACK} = \#Bit_{Präambel} + \#Bit_{Header} + t_{MAC-Header} + t_{TCPHeader}$$

$$= 72Bit + 48Bit + 272Bit + 348Bit = \underline{740Bit}$$

Nun kann mit Hilfe der in der Nebenrechnung ermittelten Werte die übermittelte Datenmenge bestimmt werden.

$$\#Bit_{ÜBest} = \#Bit_{ACKFrame} + \#Bit_{TCPACK} + \#Bit_{ACKFrame}$$

$$= 242Bit + 740Bit + 242Bit = \underline{1224Bit}$$

Die Zeit, die für die komplette Übertragung benötigt wird, kann wie nachfolgend dargestellt bestimmt werden.

$$t_{ges} = t_{ÜDaten} + t_{ÜBest} = 382{,}74\mu s + 490{,}44\mu s = \underline{814{,}22\mu s}$$

In der Abbildung 12 ist der Ablauf des Versands der einzelnen Objekte dargestellt. Auf der Empfangsseite wird immer die berechnete Dauer der Übertragung des Objektes angegeben.

Damit die Übersichtlichkeit der Darstellung erhalten bleibt, erfolgte nur für das Senden der Daten vom Sender zum Empfänger eine Darstellung der einzelnen Bestandteile. Für die nachfolgenden Empfangsbestätigungen wurden die einzelnen Bestandteile in den vorangegangenen Berechnungen angegeben. In der Abbildung 12 werden diese Bestätigungen ohne die Einzelbestandteile dargestellt.

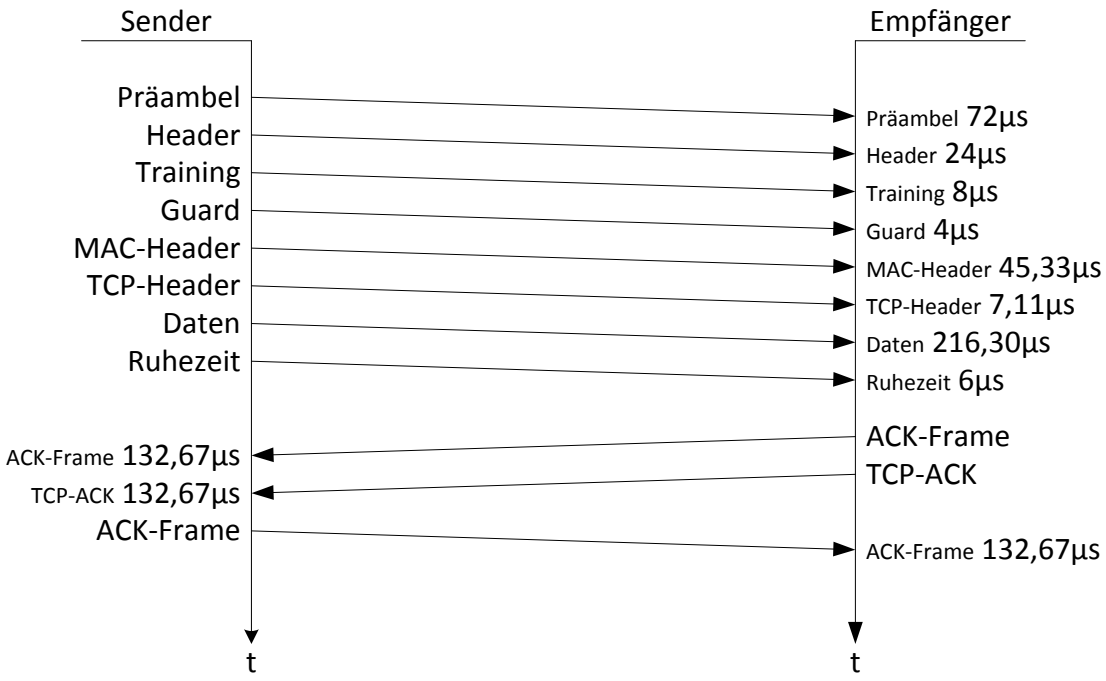

Abbildung 12: Ablauf des Datenversands

Während dieser Zeit wird die folgende Datenmenge übertragen.

$$\#Bit_{ges} = \#Bit_{\text{ÜDaten}} + \#Bit_{\text{ÜBest}} = 12456Bit + 1224Bit$$

$$= \underline{13680Bit}$$

Aus den nun ermittelten Werten kann die Datenübertragungsgeschwindigkeit ermittelt werden. Zuerst wird der Wert für alle übertragenen Daten ermittelt.

$$v_{Daten} = \frac{\#Bit_{ges}}{t_{ges}} = \frac{13680Bit}{814{,}22\mu s} = \underline{16{,}8MBit/s}$$

Nun wird der Wert für die übertragenen Nutzdaten ermittelt.

$$v_{Nutzdaten} = \frac{\#Bit_{ges}}{t_{ges}} = \frac{11680Bit}{814{,}22\mu s} = \underline{14{,}35MBit/s}$$

Für die weitere Betrachtung wird der Geschwindigkeitswert 16,8 Megabit je Sekunde herangezogen, da alle empfangenen Daten von dem Drahtlosnetzwerk-Modul an den STM32 Nucleo weitergeleitet werden sollen.

4.1.2 Datenübertragung über die UART-Schnittstelle

Auf Grund der in Kapitel 4.1.1 berechneten Übertragungsgeschwindigkeit muss das Drahtlosnetzwerk-Modul ESP8266 in der Lage sein, mindestens 17 Megabit je Sekunde über die Schnittstelle an den STM32 Nucleo weiterzuleiten.

Das hier verwendete Drahtlosnetzwerk-Modul ESP8266 kann, wie bereits in Kapitel 3.1.2 beschrieben, lediglich über eine UART-Verbindung mit dem STM32 Nucleo verbunden werden. In Verbindung mit der aktuellen Firmwareversion des Drahtlosnetzwerk-Moduls beträgt die maximal mögliche Verbindungsgeschwindigkeit über die serielle Schnittstelle 115200 Baud (vergleiche Kapitel 3.3.3.2).

Im Falle serieller Verbindungen ohne Modulationen kann die Datenübertragungsgeschwindigkeit wie folgt ermittelt werden:

$$v_{brutto} = 115200Baud = 115200Bit/s = \underline{0{,}1152MBit/s}$$

Die Übertragungszeit der vom Drahtlosnetzwerk-Modul empfangenen 13680 Bit über die serielle Verbindung kann wie folgt berechnet werden.

$$t_{Daten} = \frac{13680Bit}{115200Bit/s} = 0{,}11875s = \underline{118750\mu s}$$

Auch bei der seriellen Übertragung kommen weitere Bit hinzu. Übertragen werden 8 Datenbit. Die Übertragung wird mit einem Startbit eingeleitet und mit einem Stoppbit beendet. Das bedeutet, dass für 8 Bit an Daten 10 Bit übertragen werden (siehe Abbildung 13). Die Anzahl der zu übertragenden Bit für das betrachtete Beispiel kann wie folgt bestimmt werden.

$$\#Bit_{ges} = 13680Bit + \left(2 \cdot \left(\frac{13680Bit}{8}\right)\right) = 13680Bit + (2 \cdot 1710Bit)$$

$$= 13680Bit + 3420Bit = \underline{17100Bit}$$

Somit müssen über die serielle Verbindung 17100 Bit gesendet werden. Die dazu benötigte Zeit wird wie nachfolgend beschrieben berechnet.

$$t_{Daten} = \frac{17100Bit}{115200Bit/s} = 0{,}1484375s = \underline{148437{,}5\mu s}$$

Die Abbildung 13 verdeutlicht am Beispiel der fiktiven Bitfolge `10000110`, wie die Datenübertragung abläuft. Nach dem Stoppbit wird direkt anschließend mit einem neuen Startbit die Übertragung der nächsten acht Bit eingeleitet.

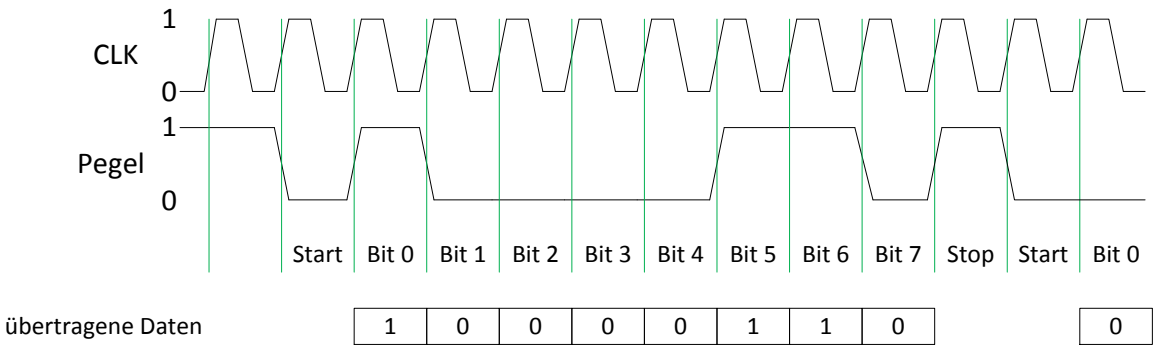

Abbildung 13: Darstellung der Übertragung über UART

Wird die Übertragungsdauer für Daten im Wireless LAN mit der Dauer der Datenübertragung der seriellen Schnittstelle verglichen, so dauert die Übertragung der Daten über die serielle Schnittstelle den Faktor 182 länger als die Übertragung der Daten über das Drahtlosnetzwerk.

Aus diesem Grund ist die serielle Schnittstelle des Drahtlosnetzwerk-Moduls ESP8266 nicht geeignet, das Drahtlosnetzwerk-Modul im Promiscous-Mode zu betreiben, da durch die Verzögerung ein zeitnahes Weiterleiten der Daten nicht möglich ist.

Weiterhin muss beachtet werden, dass auf dem Drahtlosnetzwerk-Modul genügend Speicher zum Puffern der über das Wireless LAN empfangenen und noch nicht über die serielle Schnittstelle versendeten Daten vorhanden sein muss. Außerdem müssen die Daten nach einer eventuellen Verarbeitung auf dem STM32 Nucleo erneut über eine weitere serielle Schnittstelle an das zweite Drahtlosnetzwerk-Modul übertragen werden, damit dieses die Daten weiterleiten kann.

4.1.3 Datenübertragung über die SPI-Schnittstelle

In diesem Projekt wird, wie im Kapitel 3.2.1 beschrieben, bereits das Serial Peripheral Interface (SPI) für die Verbindung mit dem SD Card Shield V4 genutzt.

Bei dem für dieses Projekt verwendeten Drahtlosnetzwerk-Modul wurden die Anschlüsse für das SPI nicht auf die PIN-Leiste geführt. Der Drahtlosnetzwerk-Chipsatz ESP8266EX, der auf dem verwendeten Drahtlosnetzwerk-Modul vorhanden ist, besitzt jedoch die benötigten Anschlüsse.

Sollen diese Anschlüsse genutzt werden, kann einerseits ein anderes Drahtlosnetzwerk-Modul, beispielsweise das Modul ESP-12E [54], verwendet werden, bei dem die entsprechenden Anschlüsse zur Verfügung stehen.

Bei einer Weiterverwendung des bereits genutzten Drahtlosnetzwerk-Moduls müssen die entsprechenden Anschlüsse nach außen geführt werden. Dazu muss sehr nahe am Drahtlosnetzwerk-Chip gelötet werden und die Abstände der Lötpunkte sind sehr gering.

In der Tabelle 9 befinden sich die benötigten PINs mit der entsprechenden Nummer und der Bezeichnung des PINs des Drahtlosnetzwerk-Chipsatzes.

Nummer des PINs des ESP8266EX	Bezeichnung des PINs	Verwendung
23	SPI_MOSI	*Master Out Slave In* Datenleitung vom Master zum Slave
24	SPI_MISO	*Master In Slave Out* Datenleitung vom Slave zum Master
21	SPI_CLK	*Clock* Taktsignal zur Synchronisation
20	SPI_CS0	*Chip Select* Signal zur Auswahl des Kommunikationspartners

Tabelle 9: Übersicht der SPI PINs des Drahtlosnetzwerk Controllers ESP8266EX

Die Übertragung von Daten mittels SPI bietet verschiedene Übertragungsgeschwindigkeiten, die nur von der Taktfrequenz abhängig sind.

Mittels Chip Select wird der entsprechende Slave ausgewählt, mit dem der Master kommunizieren möchte. Wie in der Abbildung 14 zu sehen ist, wird das Chip Select-Signal dafür auf Low gezogen und anschließend wird mit jedem Takt ein Bit eingelesen.

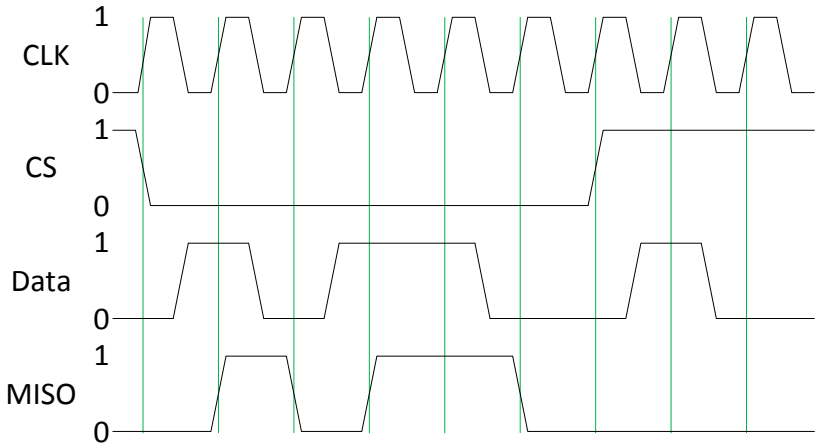

Abbildung 14: Funktionsweise SPI

Für die Datenübertragungsgeschwindigkeit ist von großem Vorteil, dass SPI keine Fehlerkontrolle durch Hardware implementiert. Das bedeutet, es werden keine zusätzlichen Paritätsbits oder ähnliches eingefügt. Ist eine Fehlerkontrolle gewünscht, muss diese also softwareseitig implementiert werden.

Da die Steuerung des Slave über eine getrennte Leitung durchgeführt wird, ist die Übertragungsgeschwindigkeit direkt von der Taktfrequenz abhängig. Je Takt kann genau ein Bit übertragen werden. Der STM32 Nucleo und der Drahtlosnetzwerk-Chipsatz ESP8266EX bieten für SPI eine Taktfrequenz von 20 Megahertz. Daraus lässt sich die Übertragungsgeschwindigkeit wie folgt bestimmen.

$$v_{Daten} = \underline{20MBit/s}$$

Es ist möglich, über das Serial Peripheral Interface Daten gleichzeitig zu empfangen und zu senden. Die Nutzung des Full Duplex Modus ist möglich, da eine Kommunikation über den MISO-PIN vom Slave zum Master und über den MOSI-PIN vom Master zum Slave erfolgt. Allerdings kann immer nur mit dem mittels Chip Select ausgewählten Slave kommuniziert werden.

Der STM32 Nucleo stellt insgesamt fünf Serial Port Interfaces zur Verfügung. Für die Umsetzung dieses Projektes würden insgesamt drei Serial Peripheral Interfaces, zwei für die Drahtlosnetzwerk-Module und eins für das SD Card Shield V4, belegt werden.

4.1.4 Zusammenfassung

Bei den in den Kapiteln 4.1.1, 4.1.2 und 4.1.3 ermittelten Werten, insbesondere bei den in Kapitel 4.1.1 ermittelten Werten für Drahtlosnetzwerke, handelt es sich um theoretisch erreichbare Werte, die die unterschiedlichen Übertragungsgeschwindigkeiten beim Aufbau der Verbindung sowie den benötigten Overhead für die Protokolle berücksichtigen. Dabei ist zu beachten, dass die Betrachtung der Qualität des Übertragungsweges nicht berücksichtigt wird. Besonders bei Drahtlosnetzwerken müssen weitere Faktoren wie Entfernung zum Funkpartner, mögliche Dämpfungen des Funksignals durch Decken oder Wände, das Zusammenspiel der Funkkomponenten untereinander und mögliche Probleme der Treiber für die Drahtlosnetzwerk-Module betrachtet werden. Bei drahtgebundenen Übertragungen können beispielsweise nicht geschirmte oder zu lange Kabel zu Problemen bei der Übertragung führen.

So wurden beispielsweise durch Madhu Vinod Diwakar und Divya Pathiramanna in der Veröffentlichung „A survey on 802.11n over 802.11g“ [55] sechs verschiedene Wireless LAN Access Points mit drei verschiedenen Wireless LAN Modulen betrieben. Bei dieser Untersuchung wurden auch die Übertragungsgeschwindigkeiten für eine TCP-Verbindung in Drahtlosnetzwerken nach dem Standard IEEE 802.11g ermittelt. Im Mittel konnte dabei eine Übertragungsgeschwindigkeit von 22,7 Megabit je Sekunde erreicht werden. Je nach Hardwarekombination lagen die Übertragungsgeschwindigkeiten zwischen 6,9 Megabit je Sekunde und 27,1 Megabit je Sekunde.

Zu ähnlichen Ergebnissen kommen Sandra Sendra, Pablo Fernandez, Carlos Turro und Jaime Lloret in ihrer Abhandlung „IEEE 802.11a/b/g/n Indoor Coverage and Performance Comparison“ [56]. In dieser Abhandlung wurden die Standards IEEE 802.11a, IEEE

802.11b, IEEE 802.11g und IEEE 802.11n miteinander verglichen. Dazu wurden verschiedene Wireless LAN Access Points der unterschiedlichen Standards verglichen. Dabei wurde für die zwei zum Standard IEEE 802.11g kompatiblen Wireless LAN Access Points für eine TCP-Verbindung ein Durchsatz zwischen 21 Prozent und 37 Prozent gemessen. Diese Werte entsprechen einer Übertragungsgeschwindigkeit zwischen 11,3 Megabit je Sekunde und 20 Megabit je Sekunde.

Betrachtet man die Ergebnisse der beiden Abhandlungen und vergleicht diese mit dem in Kapitel 4.1.1 ermittelten Wert von 16,8 Megabit je Sekunde, stellt man fest, dass dieser Wert im Mittel der in [55] und [56] veröffentlichten Werte liegt.

Betrachtet man die in den Veröffentlichungen [55] und [56] angegebenen maximalen Übertragungsgeschwindigkeiten an, stellt man fest, dass diese über den Datenübertragungsgeschwindigkeiten liegen, die über das Serial Peripheral Interface erreicht werden können. Hier besteht die Gefahr, dass im Falle eines hohen Datenaufkommens die Daten nicht komplett weitergeleitet werden können, da weder auf dem Drahtlosnetzwerk-Modul noch auf dem STM32 Nucleo ausreichend Arbeitsspeicher zur Verfügung steht.

Weiterhin muss beachtet werden, dass bei den Berechnungen bisher immer davon ausgegangen wurde, dass ein Endgerät mit dem Drahtlosnetzwerk-Modul verbunden ist. Die zu verarbeitende Datenmenge kann sich jedoch erhöhen, wenn weitere Endgeräte sich mit dem Drahtlosnetzwerk-Modul verbinden. In diesem Falle würde die Verbindung über das Serial Peripheral Interface nicht genügend Daten übertragen können. Dies kann dazu führen, dass die Daten entweder zu spät oder überhaupt nicht weitergeleitet werden können. Dadurch kann nicht sichergestellt werden, dass die aufgezeichneten Daten korrekt sind.

Aus den bisher gewonnenen Erkenntnissen muss davon ausgegangen werden, dass die Übertragungsleistung des Serial Peripheral Interfaces zu gering ist, falls nicht sichergestellt werden kann, dass sich nicht mehr als ein Endgerät mit dem Drahtlosnetzwerk-Modul verbindet. Aber auch wenn sichergestellt werden kann, dass sich nur ein Endgerät verbindet, kann der Fall eintreten, dass Datenübertragungsgeschwindigkeiten von mehr als 20 Megabit je Sekunde erreicht werden. In diesem Fall müssten die Daten zwischengespeichert werden und dann zeitversetzt versendet werden. Falls der Speicher des Drahtlosnetzwerk-Moduls oder des STM32 Nucleo dabei überläuft, würden die restlichen Daten verworfen werden.

4.2 Ausblick

Im folgenden Abschnitt werden mit der Aktualisierung der Uhrzeit und der Vorstellung alternativer Möglichkeiten zur Stromversorgung zwei wichtige und sinnvolle zukünftige Erweiterungsmöglichkeiten betrachtet.

4.2.1 Aktualisierung der Uhrzeit

Für eine konsistente und nachvollziehbare Aufzeichnung der Daten ist es notwendig, die Daten ordnungsgemäß zu protokollieren. Dazu gehört, den Startzeitpunkt der Aufzeichnung, den Empfangszeitpunkt der Pakete sowie den Zeitpunkt des Abschlusses der Aufzeichnung genau zu erfassen.

Da die Uhrzeit des STM32 Nucleo bei einer Unterbrechung der Stromversorgung immer auf den Zeitpunkt 01.01.1900 00:00 Uhr gesetzt wird, wird die interne Uhr in diesem Projekt mit Hilfe einer Datei auf der SD-Karte auf einen im Vorfeld frei definierbaren Startwert gesetzt.

Alternativ kann die Uhrzeit mit einem DCF-Modul gestellt werden. Für diesen Zweck existieren DCF-Empfänger, beispielsweise [57]. Dieses Modul empfängt das Zeitsignal DCF77 [58] aus Mainflingen bei Frankfurt am Main, das von der Physikalisch-technischen Bundesanstalt betrieben wird. Bedingt durch die Übertragung der Zeitsignale benötigt die Dekodierung der Zeit einige Minuten. Aus diesem Grund steht die aktuelle Zeit nicht sofort nach dem Starten des STM32 Nucleo zur Verfügung.

Eine weitere Möglichkeit besteht in der Nutzung einer Zeitquelle aus dem Internet, die das Zeitsignal über das sogenannte Protokoll NTP (Network Time Protocol) [59] bereitstellt. Die Physikalisch-technische Bundesanstalt stellt für diesen Zweck zwei Zeitserver zur Verfügung [60]. Damit mit Hilfe von NTP die Uhrzeit gestellt werden kann, muss eine Verbindung mit dem Internet hergestellt worden sein. Wird diese Verbindung über das Drahtlosnetzwerk hergestellt, würde dies jedoch dem zu Beginn definierten Ziel widersprechen, im Zielnetzwerk unsichtbar zu bleiben. Als Alternative könnte eine Internetverbindung über Mobilfunk hergestellt werden.

Mit den beiden bisher vorgestellten Möglichkeiten ist es ebenfalls möglich, die Uhrzeit im laufenden Betrieb zu synchronisieren und somit die Abweichung der Uhrzeit zu minimieren.

Falls im Falle eines Ausfalls der Stromversorgung die Funktionstüchtigkeit des STM32 Nucleo nicht sichergestellt sein muss, würde es ausreichen, die interne Uhr mit einer Batterie zu puffern. Dazu kann beispielsweise eine Knopfzelle genutzt werden, die an die PINs Masse (`GND`) und `VBAT` angeschlossen wird. Durch die Batterie wird die interne Uhr des Prozessors mit Strom versorgt und kann weiter laufen.

Ist es notwendig, dass der STM32 Nucleo auch bei einem Verlust der Spannungsversorgung weiterhin funktionstüchtig ist, werden die nötigen Maßnahmen im Kapitel 4.2.2 vorgestellt.

4.2.2 Stromversorgung

Ist es notwendig, dass der STM32 Nucleo auch bei einem Verlust der Spannungsversorgung weiterhin funktionstüchtig ist, kann dies beispielsweise mit einer sogenannten Powerbank gewährleistet werden. Diese Powerbanks stehen mittlerweile in verschiedenen Leistungsstufen zur Verfügung.

Zur Abschätzung des Strombedarfs der Hardware müssen die einzelnen Komponenten betrachtet werden.

- Der STM32 Nucleo benötigt bei einer Stromversorgung über den USB-Port (5 Volt) maximal 300 Milliampere [61].
- Das SD Card Shield V4 benötigt maximal 200 Milliampere bei 5,5 Volt [12].
- Ein Drahtlosnetzwerk-Modul ESP8266 benötigt maximal 170 Milliampere bei 3,3 Volt [62].

Aus den oben angegebenen Werten lässt sich der maximale Leistungsbedarf des kompletten Aufbaus wie folgt berechnen.

$$P_{ges} = P_{Nucleo} + P_{SDCardShield} + 2 \cdot P_{ESP8266}$$

$$= U_{Nucleo} \cdot I_{Nucleo} + U_{SDCardShield} \cdot I_{SDCardShield} + 2 \cdot (U_{ESP8266} \cdot I_{ESP8266})$$

$$= 5V \cdot 300mA + 5{,}5V \cdot 200mA + 2 \cdot (3{,}3V \cdot 170mA)$$

$$= 1500mW + 1100mW + 2 \cdot 561mW$$

$$= \underline{3722mW}$$

Der maximale Leistungsbedarf liegt bei 3722 Milliwatt. Bei den zur Berechnung verwendeten Werten handelt es sich um Maximalwerte, im normalen Betrieb sind niedrigere Werte zu erwarten.

Nur im Datenblatt zum ESP8266 finden sich mehrere Angaben zur Strombedarf des Drahtlosnetzwerk-Moduls. Der oben angegebene Maximalwert von 170 Milliampere gilt für das Senden von Daten, beim Empfangen verbraucht das Drahtlosnetzwerk-Modul ESP8266 mit maximal 56 Milliampere rund zwei Drittel weniger, wird weder gesendet noch empfangen benötigt das Drahtlosnetzwerk-Modul mit 15 Milliampere weniger als ein Zehntel des maximalen Wertes.

Da für die anderen Komponenten keine anderen Werte für den Leistungsbedarf angegeben sind und der Leistungsbedarf mit 3,722 Watt für ein Gerät dieser Größenordnung etwas hoch erscheint, wurde der Leistungsbedarf des gesamten Aufbaus mit einem USB-Messgerät ermittelt.

Das Messgerät wird an den USB Anschluss des Computers angeschlossen. Das zu messende Gerät wird an die USB Schnittstelle des Messgeräts angeschlossen. Das Messgerät zeigt die aktuelle Spannung, die aktuelle Stromstärke sowie den kumulierten Arbeitswert in Milliamperestunden auf dem Display an.

Zur Durchführung der Messung wurde das Programm zur Demonstration des Sendens und Empfangens so geändert, dass die Drahtlosnetzwerk-Module für eine Stunde kontinuierlich senden und empfangen. Weiterhin werden die empfangenen Daten auf die SD-Karte geschrieben.

Nach dem Ablauf des Programms wurde der folgende Arbeitswert vom Messgerät abgelesen.

$$Q = 174{,}6mAh$$

Während dieser Messung wurden die Angaben für Spannung und Stromstärke regelmäßig überwacht. Dabei sind die folgenden minimalen und maximalen Werte abgelesen worden.

$$U_{min} = 4{,}95V \qquad U_{max} = 4{,}98V$$

$$I_{min} = 165mA \qquad I_{max} = 182mA$$

Die vom Hersteller für das Messgerät angegebenen Messgenauigkeiten betragen für die Messung der Spannung ±0,1 Volt und ±20 Milliampere für die Messung der Stromstärke.

Aus den beiden Maximalwerten für Spannung und Stromstärke kann der benötigte Leistungsbedarf ermittelt werden. Da dieser, wie auch die anderen berechneten Werte dazu dienen soll, die Laufzeit beim Betrieb mit einer Powerbank abzuschätzen, werden immer die maximalen Werte verwendet. Dadurch wird sichergestellt, dass die berechnete Überbrückungszeit nicht zu großzügig berechnet wird.

$$P_{gemessen} = U_{max} \cdot I_{max}$$

$$= 4{,}98V \cdot 182mA$$

$$= \underline{906mW}$$

Zurzeit (Dezember 2016) existieren im Endverbraucherbereich Powerbanks mit einer Kapazität bis zu 18000 Milliamperestunden. Für die folgenden drei gebräuchlichsten Kapazitäten (10000 Milliamperestunden, 12000 Milliamperestunden und 18000 Milliamperestunden) werden mit dem maximalen Messwert für die Stromstärke nachfolgend die maximal möglichen Laufzeiten berechnet.

$$t_{10000mAh} = \frac{10000mAh}{182mA} \approx \underline{54h} \approx \underline{2{,}2Tage}$$

$$t_{12000mAh} = \frac{12000mAh}{182mA} \approx \underline{65h} \approx \underline{2{,}7Tage}$$

$$t_{18000mAh} = \frac{18000mAh}{182mA} \approx \underline{98h} \approx \underline{4{,}1Tage}$$

Somit ist es möglich, einen Zeitraum von mindestens 54 Stunden zu überbrücken, wenn eine Powerbank eingesetzt wird, die eine Kapazität von mindestens 10000 Milliamperestunden besitzt.

Wichtig im Falle des Betriebs mit einer mobilen Stromquelle ist, dass die Spannungsquelle überwacht werden kann, damit beispielsweise ihr Erschöpfen erkannt werden kann und die Protokollierung ordnungsgemäß beendet werden kann.

Außerdem ist es möglich, dass sich die Stromquelle nach dem Abschalten des STM32 Nucleo sowie dessen Peripherie wieder soweit erholt, dass eine erneute Stromversorgung möglich wird. Dies könnte unter Umständen zur Korruption der bisher erhaltenen Daten führen.

Literatur

Literatur

[52] J. Rech, Wireless LANs - 802.11-WLAN-Technologie und praktische Umsetzung im Detail, 3., aktualisierte und erweiterte Auflage ed., Hannover: Heise Zeitschriften Verlag GmbH & Co. KG, 2008.

[63] M. Hofherr, WLAN-Sicherheit - Professionelle Absicherung von 802.11-Netzen, Hannover: Heise Zeitschriften Verlag GmbH & Co. KG, 2005.

[64] Zheng, Zhao, Tipper, Tatuya, Shima, Qian, Peterson, Ni, Manjunath, Li, Kuri, Kumar, Krishnamurthy, Guibas, Garg, Farrel and Davie, Wireless Networking Complete, Amsterdam, Boston, Heidelberg, London, New York, Oxford, Paris, San Diego, San Francisco, Singapore, Sydney, Tokio: Morgan Kaufman Publishers by Elsevier, 2010.

[65] E. Eren and K.-O. Detken, Mobile Security - Risiken mobiler Kommunikation und Lösungen zur mobilen Sicherheit, München, Wien: Carl Hanser Verlag, 2006.

Internetquellen

[1] Raspberry Pi Foundation, „Raspberry Pi - Teach, Learn and Make with Raspberry Pi,“ [Online]. Link: https://www.raspberrypi.org/. [Zugriff am 15.08.2016].

[2] BeagleBoard.org Foundation, „BeagleBoard.org - community supported open hardware computers for making,“ [Online]. Link: https://beagleboard.org/. [Zugriff am 15.08.2016].

[3] Arduino LLC, „Arduino - Home,“ [Online]. Link: https://www.arduino.cc/. [Zugriff am 15.08.2016].

[4] STMicroelectronics, „STM32 MCU NUCLEO,“ [Online]. Link: http://www.st.com/stm32nucleo. [Zugriff am 30.07.2016].

[5] STMicroelectronics, „STMicroelectronics,“ [Online]. Link: ww.st.com. [Zugriff am 30.07.2016].

[6] STMicroelectronics, „STM32F411RET6,“ [Online]. Link: http://www2.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f4-series/stm32f411/stm32f411re.html. [Zugriff am 30.07.2016].

[7] STMicroelectronics, „STM32 NUCLEO Product Specifications,“ [Online]. Link: http://www2.st.com/resource/en/databrief/nucleo-f030r8.pdf. [Zugriff am 30.07.2016].

[8] mbed, „mbed developers,“ [Online]. Link: https://developer.mbed.org/platforms/ST-Nucleo-F401RE/#board-pinout. [Zugriff am 30.07.2016].

[9] Espressif Systems, „ESP8266EX Ressources,“ [Online]. Link: http://www.espressif.com/en/products/hardware/esp8266ex/resources. [Zugriff am 01.10.2016].

[10] Espressif Systems, [Online]. Link: http://www.espressif.com. [Zugriff am 01.10.2016].

[11] Espressif Systems, „ESP8266EX Datasheet,“ [Online]. Link: http://www.espressif.com/en/file/397/download?token=oVtCENmg. [Zugriff am 13.10.2016].

[12] Seeed Development Limited, „SD Card Shield V4,“ [Online]. Link: http://www.seeedstudio.com/SD-Card-Shield-V4-p-1381.html. [Zugriff am 01.12.2016].

[13] ePanorama.net, „Serial buses information page,“ [Online]. Link: http://www.epanorama.net/links/serialbus.html#spi. [Zugriff am 12.11.2016].

[14] Franklin F. Kuo, University of Hawaii, Honolulu, „Computer Networks - The ALOHA-System,“ 11.08.1981. [Online]. Link:

http://nvlpubs.nist.gov/nistpubs/jres/086/jresv86n6p591_A1b.pdf. [Zugriff am 21.10.2016].

[15] IEEE, „Working Group for WLAN Standards," [Online]. Link: http://grouper.ieee.org/groups/802/11/. [Zugriff am 23.08.2016].

[16] IEEE, „IEEE 802.11 Standard," 1997. [Online]. Link: http://standards.ieee.org/findstds/standard/802.11-1997.html. [Zugriff am 26.10.2016].

[17] IEEE, „IEEE 802.11a Standard," 1999. [Online]. Link: http://standards.ieee.org/findstds/standard/802.11a-1999.html. [Zugriff am 26.10.2016].

[18] IEEE, „IEEE 802.11b Standard," 1999. [Online]. Link: http://standards.ieee.org/findstds/standard/802.11b-1999.html. [Zugriff am 26.10.2016].

[19] IEEE, „IEEE 802.11g Standard," 2003. [Online]. Link: http://standards.ieee.org/findstds/standard/802.11g-2003.html. [Zugriff am 26.10.2016].

[20] IEEE, „IEEE 802.11n Standard," 2009. [Online]. Link: http://standards.ieee.org/findstds/standard/802.11n-2009.html. [Zugriff am 26.10.2016].

[21] IEEE, „IEEE 802.11ac Standard," 2013. [Online]. Link: http://standards.ieee.org/getieee802/download/802.11ac-2013.pdf. [Zugriff am 26.10.2016].

[22] Elektronik Kompendium, „IEEE 802.11ac / Gigabit-WLAN," [Online]. Link: http://www.elektronik-kompendium.de/sites/net/1602101.htm. [Zugriff am 23.10.2016].

[23] E. Tews, R.-P. Weinmann und A. Pyshkin, „Breaking 104 bit WEP in less than 60 seconds," 2007. [Online]. Link: http://eprint.iacr.org/2007/120.pdf. [Zugriff am

23.09.2016].

[24] IEEE, „IEEE 802.11i Standard," 2004. [Online]. Link: http://standards.ieee.org/findstds/standard/802.11i-2004.html. [Zugriff am 26.10.2016].

[25] Wireshark, „WLAN (IEEE 802.11) capture setup," [Online]. Link: https://wiki.wireshark.org/CaptureSetup/WLAN#Promiscuous_mode. [Zugriff am 12.11.2016].

[26] Bundesministerium der Justiz und für Verbraucherschutz, „Strafgesetzbuch," [Online]. Link: https://www.gesetze-im-internet.de/stgb/index.html. [Zugriff am 11.10.2016].

[27] heise Security, Stefan Krempl, „Verschärfte Hackerparagraphen treten in Kraft," 10.08.2007. [Online]. Link: http://www.heise.de/security/meldung/Verschaerfte-Hackerparagraphen-treten-in-Kraft-162011.html. [Zugriff am 11.09.2016].

[28] heise Security, Daniel Bachfeld, „Vorkonfigurierte WPA-Schlüssel bei T-Online und Vodafone leicht erratbar," 20.08.2011. [Online]. Link: http://www.heise.de/security/meldung/Vorkonfigurierte-WPA-Schluessel-bei-T-Online-und-Vodafone-leicht-erratbar-1326796.html. [Zugriff am 11.09.2016].

[29] heise Security, Ronald Eikenberg, „Router-Schwachstelle für Telefonterror missbraucht," 21.08.2013. [Online]. Link: http://www.heise.de/security/meldung/Router-Schwachstelle-fuer-Telefonterror-missbraucht-1939225.html. [Zugriff am 11.10.2016].

[30] S. Viehböck, „Brute forcing Wi-Fi Protected Setup," 26.11.2011. [Online]. Link: http://sviehb.files.wordpress.com/2011/12/viehboeck_wps.pdf. [Zugriff am 11.09.2016].

[31] Espressif Systems, „AT Firmware," [Online]. Link: https://github.com/espressif/esp8266_at. [Zugriff am 09.08.2016].

[32] S. Schocke, „Updating ESP8266 Firmware," [Online]. Link: https://developer.mbed.org/users/sschocke/code/WiFiLamp/wiki/Updating-ESP8266-

Firmware. [Zugriff am 29.06.2016].

[33] S. Schocke, „Flasher ES8266,“ [Online]. Link: https://developer.mbed.org/media/uploads/sschocke/esp_flasher.zip. [Zugriff am 29.10.2016].

[34] tcp-ip-info.de - Gerhard M. Glaser, „Hayes Befehlssatz und andere Modemparameter,“ [Online]. Link: http://www.tcp-ip-info.de/tcp_ip_und_internet/hayes-befehle.htm. [Zugriff am 13.11.2016].

[35] Arduino, „A Brief Introduction to the Serial Peripheral Interface (SPI),“ Arduino, [Online]. Link: https://www.arduino.cc/en/Reference/SPI. [Zugriff am 09.08.2016].

[36] microcontroller.net, „Serial Peripheral Interface,“ [Online]. Link: http://www.mikrocontroller.net/articles/Serial_Peripheral_Interface. [Zugriff am 08.09.2016].

[37] mbed, „SDFileSystem,“ [Online]. Link: https://developer.mbed.org/teams/mbed/code/SDFileSystem/. [Zugriff am 01.08.2016].

[38] opengroup.org, „fopen,“ [Online]. Link: http://pubs.opengroup.org/onlinepubs/9699919799/functions/fopen.html. [Zugriff am 15.08.2016].

[39] opengroup.org, „fclose,“ [Online]. Link: http://pubs.opengroup.org/onlinepubs/9699919799/functions/fclose.html. [Zugriff am 15.08.2016].

[40] opengroup.org, „fprintf,“ [Online]. Link: http://pubs.opengroup.org/onlinepubs/9699919799/functions/fprintf.html. [Zugriff am 15.08.2016].

[41] opengroup.org, „fgetc,“ [Online]. Link: http://pubs.opengroup.org/onlinepubs/9699919799/functions/fgetc.html. [Zugriff am 15.08.2016].

[42] mbed.org, „mbed.org Developer,“ [Online]. Link: http://developer.mbed.org. [Zugriff am 10.11.2016].

[43] mbed.org, „mbed.org Compiler,“ [Online]. Link: https://developer.mbed.org/compiler. [Zugriff am 10.11.2016].

[44] Espressif Systems, „SDK & Demos,“ [Online]. Link: http://espressif.com/en/support/download/sdks-demos?keys=&field_type_tid%5B%5D=14. [Zugriff am 13.11.2016].

[45] Espressif Systems, „SDK Environment,“ [Online]. Link: https://drive.google.com/folderview?id=0B5bwBE9A5dBXaExvdDExVFNrUXM&usp=sharing. [Zugriff am 01.11.2016].

[46] Espressif Systems, „SDK Getting Started Guide,“ [Online]. Link: http://espressif.com/sites/default/files/documentation/2a-esp8266-sdk_getting_started_guide_en.pdf. [Zugriff am 01.11.2016].

[47] ITEAD STUDIO, „WeeESP8266,“ [Online]. Link: https://developer.mbed.org/users/itead/code/WeeESP8266/. [Zugriff am 03.11.2016].

[48] ITEAD STUDIO, „ArduinoAPI,“ [Online]. Link: https://developer.mbed.org/users/itead/code/ArduinoAPI/. [Zugriff am 03.11.2016].

[49] room-15, „ESP8266 - AT Command Reference,“ [Online]. Link: https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/. [Zugriff am 20.12.2016].

[50] ST Microelektronics, „ST-LINK,“ [Online]. Link: http://www.st.com/en/embedded-software/stsw-link009.html. [Zugriff am 14.11.2016].

[51] Simon Tatham, „PuTTY,“ [Online]. Link: http://www.putty.org/. [Zugriff am 15.11.2016].

[53] Network Working Group of the Internet Engineering Task Force, „RFC 894 - A Standard for the Transmission of IP Datagrams over Ethernet Networks,“ [Online]. Link: https://tools.ietf.org/html/rfc894. [Zugriff am 01.01.2017].

[54] EXP-TECH, „ESP8266 SMT Module - ESP-12E,“ [Online]. Link: http://www.exp-tech.de/esp8266-smt-mobule-esp-12. [Zugriff am 12.12.2016].

[55] M. V. Diwakar und D. Pathiramanna, „A survey on 802.11n over 802.11g,“ 11.05.2011. [Online]. Link: http://sites.tech.uh.edu/isgrin/files/2013/11/Divya_Madhu_report.pdf. [Zugriff am 13 12 2016].

[56] S. Sendra, P. Fernandez, C. Turro und J. Lloret, „IEEE 802.11a/b/g/n Indoor Coverage and Performance Comparison,“ 12.05.2010. [Online]. Link: http://personales.upv.es/turro/articulos/IEEE802-11abgn.pdf. [Zugriff am 13.12.2016].

[57] Pollin Elektronc GmbH, „DCF-Empfangsmodul DCF1,“ [Online]. Link: http://www.pollin.de/shop/dt/NTQ5OTgxOTk-/Bauelemente_Bauteile/Bausaetze_Module/Module/DCF_Empfangsmodul_DCF1.html. [Zugriff am 07.12.2016].

[58] Physikalisch-technische Bundesanstalt, „PTB Mitteilungen 114 - Zeit- und Normalfrequenzverbreitung mit DCF77,“ 2014. [Online]. Link: http://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_4/4.4_zeit_und_frequenz/pdf/2004_Piester_-_PTB-Mitteilungen_114.pdf. [Zugriff am 07.12.2016].

[59] Network Time Foundation, „NTP,“ [Online]. Link: http://www.ntp.org/. [Zugriff am 07.12.2016].

[60] Physikalisch-technische Bundesanstalt, „Gibt es die Zeit auch online?,“ [Online]. Link: http://www.ptb.de/cms/ptb/fachabteilungen/abt4/fb-44/fragenzurzeit/fragenzurzeit07.html. [Zugriff am 07.12.2016].

[61] STMicroelectronics, „STM32F411RE User Manual,“ [Online]. Link: http://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM0

0105823.pdf. [Zugriff am 01.12.2016].

[62] Shenzhen Anxinke Technology CO.LTD, „ESP-01,“ [Online]. Link: http://www.watterott.com/media/files_public/npuccpmpv/AI-Thinker_ESP-01.pdf. [Zugriff am 02.12.2016].

[65] W. Alliance, „WiFi Alliance,“ [Online]. Link: http://www.wi-fi.org/. [Zugriff am 23.08.2016].

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 11.01.2017

Philipp Engler